THE BEST SELLING COMPUTER PROJECTS MAGAZINE NOVEMBER 1983

ELECTRONICS & COMPUTING

13 ORIC TAPE

ORIC TALK 14 (EXP-BOARD) 14 6502 CONTROLLER

19 28 CONTROL 36

34 EPROM PROG 66 TAPE

MICROGRAPHICS 67 72 LOGIC PULSER

74 ORIC APPS 75 REF CIRCUIT ANALYSIS

79 HARDWARE 80 CODING 87 REF

92 KIT COMP.

93 BOOKS

MONTHLY

AN EMAP PUBLICATION

Germany D5.80 U.S.A. $2.95 85p

LIE DETECTOR:
BUILD A BBC INTERFACE

THE NEW MICRO-PROFESSOR REVIEWED • USING A LOGIC PULSER

SPECTRUM vs ORIC vs DRAGON vs... THE TOP 10 REVIEWED

ZX DISCO LIGHT CONTROLLER • CIRCUIT MODELLING ON THE BBC

Electronics & Computing Monthly
Scriptor Court,
155 Farringdon Road,
London,
EC1R 3AD

EDITORIAL
01-833-0846

Editor
Gary Evans

Assistant Editor
William Owen

Administration
Liz Gregory

ADVERTISING
01-833-0531

Manager
Claire Fullerton

Deputy Manager
Alun Evans

Executive
Richard Jansz

PRODUCTION
01-0531-0846

Design
Pat Haylock

Make-up
Time Graphics

Publisher
Alfred Rolington

Distribution
EMAP
National Publications

Published By
EMAP Business And
Computer Publications

Printed by
Riverside Press,
England

Subscriptions
Electronics &
Computing Monthly,
(Subscription
Department),
Competition House,
Farndon Road,
Market Harborough,
Leicestershire.

ABC
MEMBER OF THE AUDIT
BUREAU OF CIRCULATION

Vol. 3

IN THIS ISSUE

Issue 11

Lie Detector Interface

This project can monitor a subject's heart rate and body temperature to give an indication of stress.

Speech Synthesis Systems 24

Ian Campbell begins a comprehensive examination of the various ways in which IC manufacturers have tackled the problem of speech synthesis.

Single Chip Microcontroller 28

Part two of this project describes the construction of the 68705 controller and continues with the description of the device's features.

MPF-1 Plus Review 32

This upgraded version of the popular Microprofessor offers a considerably improved performance when compared to the older design.

BBC EPROM Programmer 37

This month the software to control what has become our most popular project to date.

ZX Disco Light Controller 44

This project allows either a ZX81 or Spectrum to control eight banks of lights. Sophisticated features such as an auto zero crossing detector add to the appeal of the design.

Top 10 Computers Reviewed 49

We take a look at the best selling home computers and compare and contrast the features offered by the different machines.

FORTH Compilers 64

We look at a range of packages that allow FORTH programming on a range of popular micros.

Computer Graphic Techniques 67

Mike James contines his series that explores the world of computer generated graphics.

Using A Logic Pulser 72

Harry Fairhead reviews GSC's Logic Pulser kit and describes how this useful piece of test gear is used.

BBC Micro Circuit Modeller 75

A sophisticated package that can model the action of analogue circuitry.

Optimal Coding 75

James Dick concludes his look at the various ways in which data can be processed in order to make the most limited transmission channels.

Book and Software Reviews 84

Our regular look at the latest book and software publications.

Tasword 2 Reviewed 87

Word Processing and the Spectrum are rarely mentioned in the same breath, but the Tasword package succeeds in transforming Sinclair's best selling machine into a useful WP system.

Comment 9
News 13
New Products 14
Next Month 63
Subscriptions 63
Reader Services
PCB Service 81
Technical Books 93

See page 90 for details of this months free to enter competition and page 63 for details of an exclusive *E&CM* travel offer.

Both the second part of the Hi-Res computer's A/D board and of our series Practical Electronic Techniques have been held over due to lack of space.

Electronics & Computing Monthly is normally published on the 13th day of each month

EDITORIAL

Currah's μspeech synthesis system for the Spectrum arrived at our offices too late for a full review to be prepared for this issue but many of you will have had a chance to see the device at the PCW show. The device is interesting not only for the fact that it offers one of the most sophisticated speech generation systems available but also because it incorporates a ULA.

The ULA is a familiar component of computers like the Spectrum, ZX81 and Acorn's new Electron but until now use of this particular breed of IC has been restricted. The μspeech represents one of the first low cost perophéral devices from a company smaller than the likes of Acorn to make use of ULA technology.

The advantages are obvious. The μspeech is a small product with a sophisticated performance that retails at a very competitive price. With only three IC's and very few discreet components it should be both inexpensive to produce and reliable in operation. What's more, using a ULA means that nobody will be able to 'rip off' Currah's design.

The ULA for the μspeech was developed in conjunction with GI Microelectronics, a company that are keen to talk to other manufacturers about the possibility of using ULA technology within their product range.

Over the next few years, increasing expertise in designing and manufacturing logic arrays should lead to a stream of products offering high performance at low prices.

Overwhelmed

The project featured on our October issue's cover is destined to become our most popular ever. Within days of publication, our offices were flooded with orders for the EPROM Programmer's PCB and many people phoned up asking for a sneak look at this months concluding part of the project.

To those of you that had to wait slightly longer than usual for their PCB, we're sorry but we processed the orders as quickly as possible. To those of you who asked to see the software in advance of publication, again sorry but we just could not have afforded all those photo copies.

Your Comments Please

As well as featuring a very popular project, last month saw the first issue with our 'new look' front cover. General reaction to the re-design has been quite favourable – apart from many of you who pointed out that the EPROM was in its socket the wrong way round. We would however be pleased to hear any other comments from readers about the cover or about any other aspect of the magazine.

The letters page will be returning with our December issue and a selection of views will be printed then. We're also thinking that, in the best Fleet Street Tradition, we should offer a prize for the best letter received.

Put pen to paper and, you never know, you could be our first winner.

Gary Evans.

And now . . . Interface 2

The latest Spectrum add-on from Sinclair was launched at the Personal Computer World Show, just 2 months after the launch of the Microdrive and Interface 1.

Interface 2 is a neatly styled package for the games enthusiast. The top of the Interface includes a 'porthole' for ROM cartridges (ten programs on ROM are now available) and there are two joystick ports (which have potential for more serious applications) accepting standard 9-way D-plugs.

ROM cartridge software offers the advantages of convenience and speed in loading. In addition, all ROM cartridges will work with a 16K Spectrum, even if the equivalent title on cassette requires a 48K computer.

The device plugs directly into the Spectrum rear expansion port of ZX Interface 1. No extra interface is needed to match the joystick to the Spectrum or software, and, once connected, the joystick controls will work with either ROM cartridge, cassette or Microdrive programs. Like the Microdrive, Interface 2 will be available initially only through mail order. It is priced at £19.95 inc. VAT. A full review of Interface 2 will be included in next month's *E&CM*.

Electron Production Doubled

Demand for the Acorn Electron (reviewed in last month's *E&CM*) has enabled Acorn Computers to double production of the machine. The first batch of 100,000 units is being constructed in Malaysia, and now a contract for a further 100,000 has been awarded to AB Electronics, manufacturers of the BBC Micro.

The contract will provide 100 jobs at AB Electronics' new factory in Rogerstone, Gwent. The factory, which is being extended specifically for the Electron line, is said to have the most advanced electronics production equipment in the UK.

Acorn expect continued high sales of the new machine, confident that the Electron's compatibility with the BBC Micro will ensure it an importance in the home equal to that of the BBC in education.

Five years after the company's foundation, Acorn Computers has a turnover of £42 million, placing it in the top five UK computer companies along with ICL, Ferranti, Systime and Sinclair.

Software And The Law

The Computer Retailers Association (CRA) have issued a statement in response to a number of articles on the subject of software copyright which have appeared in the computer press.

A. J. Harding of the CRA claims that the articles were misleading, in that they assumed that the copyright laws did not cover computer software, and would not until the law was clarified over a period of some years. The CRA say this is not true, and point to three recent cases in which successful claims of breach of copyright were made in the courts by software authors.

According to the CRA, software becomes protected by law immediately it is written, and comes within the bounds of the 1956 Copyright Act as a 'literary or artistic work'.

The extent of software 'piracy' is unknown, but the practice is possibly widespread; the results of prosecution are also uncertain in that they depend very much on the attitude taken by the plaintiff. However, you have been warned!

Sinclair Boom

One company apparently immune from the financial problems besetting the computer world is Sinclair Research. On September 1st Sinclair announced pre-tax profits of £14.03 million (compared with £8.55 million in 1982), and a doubling of turnover to £54.53 million.

Writing in the Company's annual report, Clive sinclair attributed the 'strong position in the personal computer market to its (Sinclair's) technical and marketing leadership', and added that, 'there remains much scope for innovation in the field and I believe we will continue to lead the world with future products'.

Dragon Disks Survive Crisis

Despite their financial problems, Dragon Data have been able to launch the Disk Drive unit for the Dragon 32.

Priced at £275, the single 5¼" half height drive is easily expandable to a double disk system by inserting an additional drive. Two double units can be linked to form a 4 drive system. Memory capacity (formatted) is 184320 bytes on single sided double density disks. The disk operating system is held in ROM on the controller cards.

The drives should be in the shops immediately, though some doubts remain about their availability and the size of the initial order Dragon have made, which may be quite small.

The company was able to launch the drive after investors injected an extra £2.5 million of capital to stave off a financial crisis caused by falling demand.

Micronet Success

Micronet 800 has become the first Information Provider on Prestel to register over 1 million accesses in a single month.

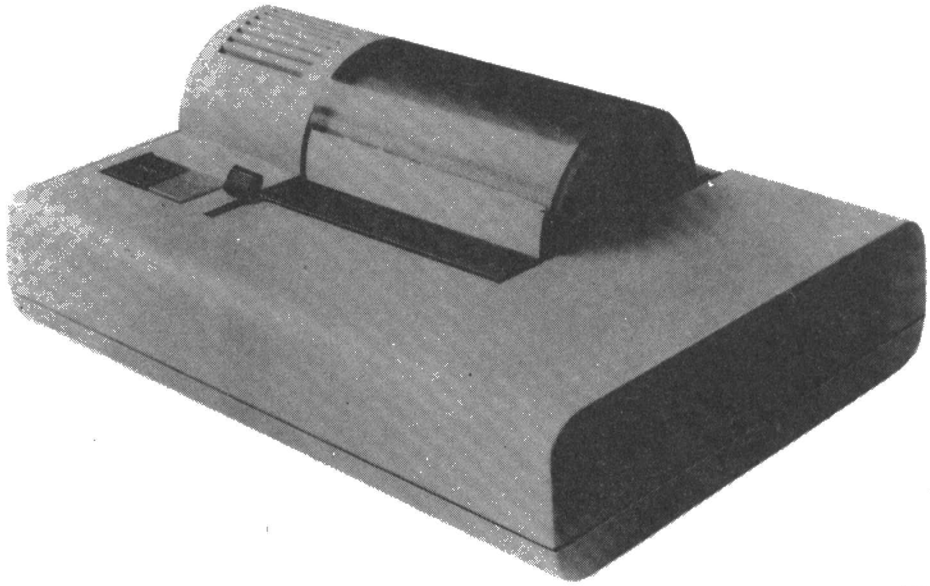
The software database system is proving increasingly popular, and this high level of interest is expected to grow as three more microcomputers were able to link up with the database from September. These are the Sinclair Spectrum (via the Prism VTX 5000 videwdata modem), the Apple II, and the Tandy TRS 80.

Oric User

Oric owners now have their own independent monthly to complement E&CA: the *Oric User*. The newstand magazine is priced at 85p, or £10.00 per year for subscribers.

Oric User,
20 Wynford House,
Wynford Road,
London N1 9Q7.

Printer For The BBC



Now available from Dean Electronics is a 40 column thermal printer with a Centronics interface module for use with the BBC Microcomputer.

The printer uses the Olivetti thermal print mechanism which eliminates the need to change ribbons. The printer has a speed of 2 lines per second: a 24 line CRT screen can be printed in just 12 seconds. Full 320 X n dot graphics are available.

The printer costs £149.50 including separate power pack and BBC Centronics interface module.

Dean Electronics,
Glendale Park,
Fernbank Road,
Ascot,
Berks.

Making The Oric Talk

A voice synthesiser designed for use with the Microtan 65 and Oric micros is now available from Kenema Associates. The Allaphone system (an extended phoneme processor) is used to give the computer an unlimited vocabulary in any language. The programmable synthesiser has an independent power supply, on board monitor speaker, and a Hi Fi connection for greater sound volume. With a manual (also supplied in foreign languages), the synthesiser is priced at £29.95 plus VAT.

Kenema Associates Ltd.,
1 Marlborough Drive,
Worle,
Avon BS22 0DQ.

Oric Expansion Board

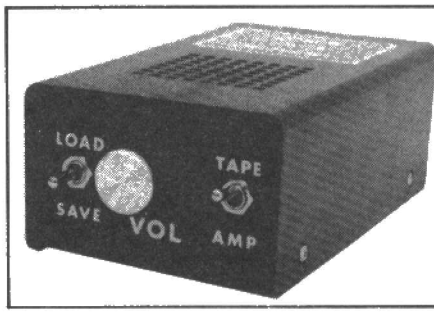
Considerable expansion of the Oric 1 is possible using a 6 slot expansion system. The mother board is the first of a new range of hardware add-ons to the Oric announced by Kenema Associates.

The board incorporates the Main Bus Expansion, Parallel Printer Bus, RGB and Cassette/Sound Bus extensions. These mother boards may be interconnected for even greater expansion and flexibility.

Meanwhile, the Software Division of Kenema has produced a new package for the Oric-1 in the form of 'Oricstar', a word processing package incorporating full screen editing, string search and replacement, full printer support including the transmission of control codes for printer operation, word wrap, document and secondary text store files, Mailshot retrieval system and more.

Kenema Associates,
1 Marlborough Drive,
Worle,
Avon BS22 0DQ.

Spectrum Cassette Controller ▶



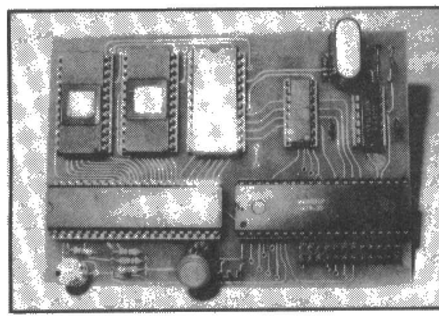
For those Spectrum users who cannot afford to pay or wait for a Microdrive, two new products are available to make program storage on cassette easier and faster.

The NMS Tape Control is a programmable cassette recorder controller which is activated by simple BASIC commands, either direct from the keyboard or under program control. One or two recorders may be plugged into the controller which carries out lead switching for LOAD and SAVE and also switches the recorders on and off via their REMOTE sockets. With one recorder set to LOAD and the other to SAVE an automatic filing and retrieval system can be set up using software such as Masterfile. A semi-automatic switch is fitted for rewind and fast forward.

The controller plugs into the Spectrum's cassette sockets leaving the expansion port free and does not use up any input/output space; it also has a built in BEEP amplifier with volume control. A kit version of the controller costs £16.95, and the fully assembled device is £19.95 (plus £1.50 p&p).

The second product from NMS is Speedyload software (48K only) enabling users to LOAD and SAVE programs at 3000 baud thus halving the waiting time for longer programs. A 1330 byte relocatable machine code program in RAMtop allows the high-speed facility to be selected with any tape command simply by preceding the command with a USR statement. No extra hardware is needed and the software can be used with a standard recorder. It costs £3.95 plus 40p p&p.

Ness Micro Systems,
100 Drakies Avenue,
Inverness IV2 3SD.



▲

6502 Micro Controller

Control 65 is a small, low cost micro controller PCB allowing stand alone terminals to have intelligence and flexibility. A +5V supply is all that is required to make the 75mm x 100mm PCB into a versatile controller offering 16TTL compatable I/O lines, up to 8K bytes of EPROM decoding, 2K bytes of user RAM plus the popular 6502 microprocessor. Onboard links allow 2716 or 2732 EPROM devices to be used. PIO interrupts are serviced for fast I/O times.

The card, which is easily programmed, is supplied with full user notes and circuit diagram at a price of £49.95.

J.P. Designs,
37 Oyster Row,
Cambridge,
CB5 8LJ.

ZX81 EPROM Reader

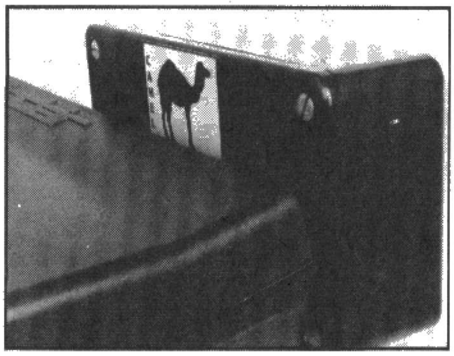
ROM-81 is a memory expansion unit for the Sinclair ZX81, which enables the user to read useful routines and commonly used information stored in EPROM.

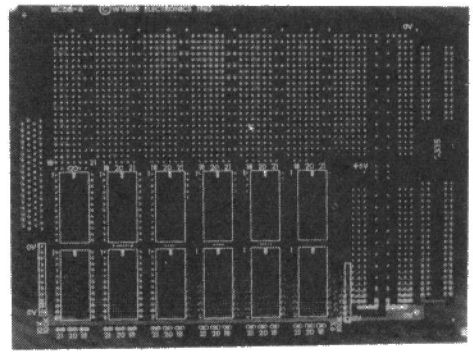
Two 24 pin sockets allow either 2716 or 2732 EPROMs to be used. They can provide up to 8Kbytes of memory in 2Kbyte increments. The sockets are decoded to lie between 8K and 16K in the ZX memory map, which is just below the BASIC area. Separate 2K and 4K decoding is link selectable to make it possible to vacate locations occupied by other peripheral cards.

A particularly useful feature of ROM-81 is additional circuitry to allow the use of slow EPROMs.

ROM-81 comes in a black ABS case with a screwed down cover. It plugs on to the ZX81 with an adaptor at the rear of the box for further expansions and is supplied with user notes which give the programs for data retrieval. The price is £14.95 plus VAT.

Cambridge Microelectronics,
1 Milton Road,
Cambridge,
CB4 1UY.





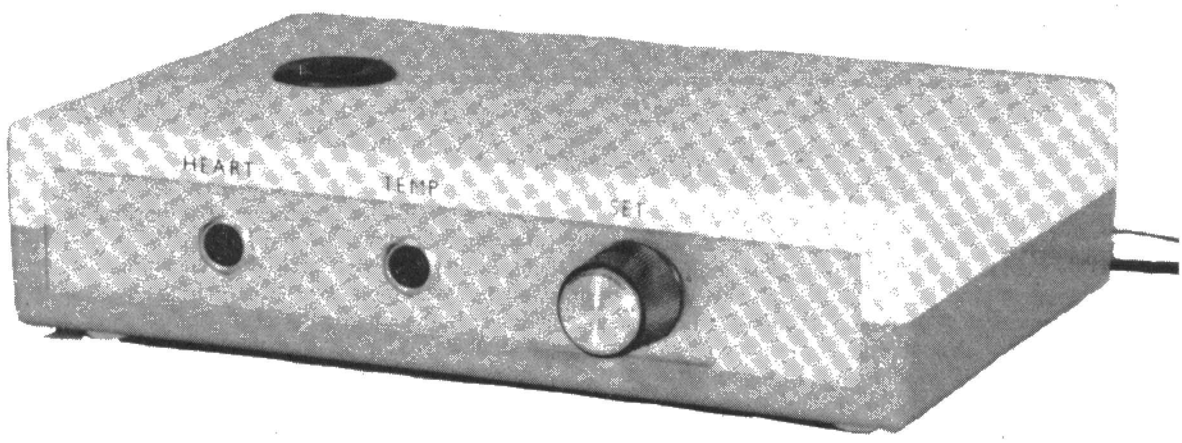
Left: the ROM-81 memory expansion unit for the ZX81, by Cambridge Microelectronics. Right: The MCDB-A; one of a range of micromputer development boards from Wymer Electronics.

Development Board

Available from Wymer Electronics are a range of Microcomputer development boards. The boards feature high quality double sided PCB layout; solder resist finish; spaces for TTL/CMOS logic; spaces for microprocessor and peripheral devices; ready-laid out interconnections for memory chips assembly; one D type and two ID connector spaces. The board's dimensions are 20cm x 15cm.

In addition the MCDB-A accepts up to twelve 24-pin SRAM/ROM/EPROM chips with only 3 control pins unconnected. The MCDB-B accepts eight 16K, 64K or 256K dynamic RAM chips and two 28/24-pin ROM/EPROM chips. Price is £14.95 inclusive.

Wymer Electronics,
8 Briar Place,
Dunfermline,
Fife KY11 4BE.



LIE DETECTOR INTERFACE

A simplified version of the polygraph machine used as a lie detector in America and finding increasing use in the UK. The interface can also be used as an aid to relaxation. Robert Penfold describes the design.

Monitoring various aspects of a body's parameters such as heart rate and skin resistance can give a good indication as to the level of stress a person is under. By providing someone with visual or oral information about the way in which their body is performing can allow them to consciously control, for example, their heartbeat and so enable them to relax.

Another use for biofeedback circuits is the much publicised polygraph. Here the monitoring takes place while a subject is under 'interrogation'. The pattern is for a series of innocent questions to be asked first – in order to establish nominal levels for the various parameters being monitored and then to ask about the areas in which the subject may be lying. Only the most accomplished liar will be able to prevent the tell-tale signs of increased stress while giving an untruthful answer to a question.

This article describes two inexpensive and straightforward biofeedback interfaces which are primarily intended for use with the BBC model B computer, but which could be used with other machines which have suitable digital and analogue input ports. The suggested software gives a graph of results on the screen of the television set or monitor, but this could easily be modified to some other form of output if desired.

Heart Rate

Measurement of heart rate is a useful form of biofeedback as this is a body function which reflects ones emotional state. Generally speaking it is relatively slow when relaxed, and increases when angry, excited, or with physical exertion.

There are several ways of electronically monitoring heart rate, but an optical system (known as photoplethysmography) is probably the one which is most practical where an easy to use piece of equipment is required. It also has the advantage that there is no direct electrical connection to the body of the user.

The sensor consists of a miniature photocell which is held above a finger nail. A fairly bright light is positioned beneath the finger-tip, and the photocell receives the light which passes through the finger-tip. Blood-flow in the capillary bed of the finger causes variations in the light level received by the photocell, and therefore results in small changes in cell resistance. These changes are very small indeed, but if highly amplified give a logic output signal at a frequency which equal to the heart rate of the user.

The Circuit

gives a short delay before the unit starts to operate properly when it is initially used.

In order to obtain an output voltage swing of a few volts peak to peak a very high voltage gain is needed. IC1b is therefore used as a non-inverting amplifier which gives a further boost in gain of around 60dB (1000 times). A rather high noise level would be produced as a consequence of this high level of gain, but C5 rolls-off the frequency response of the circuit at frequencies of more than a few tens of Hertz, and gives a substantial increase in the signal-to-noise ratio.

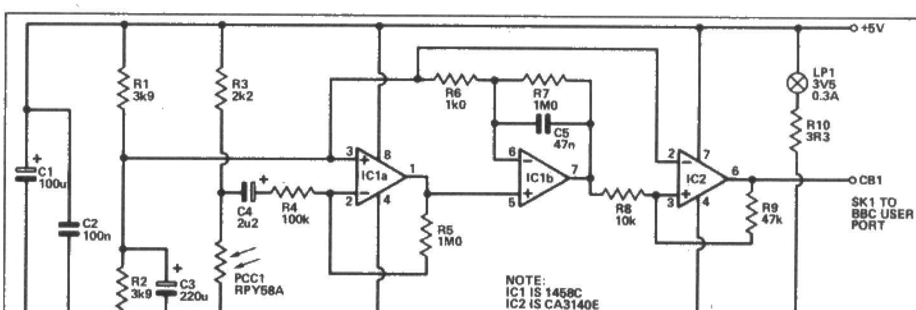


Figure 1. Full circuit diagram of the interface.

Figure 1 shows the circuit diagram of the heart rate interface.

PCC1 is the photocell sensor, and is a cadmium sulphide photoresistor. In conjuction with R3 it forms a potential divider across the supply lines so that changes in its resistance are converted to small voltage changes. These changes are coupled by C4 to an inverting operational amplifier based on IC1a and having a voltage gain of 20dB (10 times). C4 has what seems like an excessively high value by audio amplifier standards, but the frequencies involved in this application are sub-audio at about 1 to 2Hz, and C4 therefore needs a value about ten times higher than would be needed for audio use. Due to the high value of C4 it takes two or three seconds for the charge on this capacitor to take up the correct level at switch-on, or whenever PCC1 is subjected to a large change in light level. In practice this

The output signal needs to be in a form that can drive one of the logic inputs of the computer's user port. IC2 is used as a Schmitt Trigger which processes the signal to give an output which switches cleanly between the two logic levels. This is used to drive the CB1 input of the User Port. This input responds to changes in logic state rather than to a particular logic level, which is very convenient for a frequency measuring application such as the present one.

Originally the circuit was powered from a 9 volt battery, but it was found to operate perfectly well on the 5 volt output of the BBC machine, and no problems were experienced with noise on the supply rails giving spurious triggering of the circuit. The final unit is therefore fully powered from the computer, including light source LP1 which is powered from the 5 volt supply via dropper resistor R10. Although LP1 has a voltage rating of

3.5 volts, it is intended for use with a 4.5 volt battery, and requires a supply voltage of around 4 to 4.5 volts. The value of R10 has been calculated accordingly. A substantial supply current of about 0.3 amps is consumed by LP1, but the BBC computer has a large power supply that can handle this current with ease.

Temperature Sensor

On the face of it there may seem to be little correlation between body temperature and how relaxed (or otherwise) an individual happens to be. However, stress often manifests itself in the form of reduced circulation to the extremities, with a consequent drop in the temperature of ones fingers and toes. Thus, a temperature sensor attached to a finger can give a relative indication of stress, with minimum temperature corresponding to maximum stress. The changes in temperature are typically quite small, but can be readily detected using even the most basic electronic temperature sensing circuit.

The Circuit

The temperature interface uses an inexpensive (VA10665) thermistor as the sensor, and the relevant circuit diagram appears in **Fig 2.**

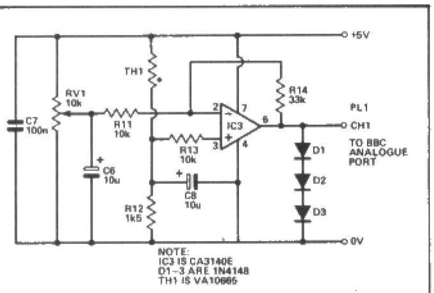


Figure 2. Circuit diagram of the Temperature sensor.

Together with R2 the thermistor forms a potential divider circuit connected across the supply lines. The VA10665 has a negative temperature coefficient (i.e. its resistance decreases if its temperature is increased), and the output voltage of the potential divider accordingly rises and falls in sympathy with increases and reductions in TH1's temperature.

IC3 is used as a non-inverting DC amplifier which boosts the output voltage swing from the sensing circuit by a factor of just over four times. The analogue input of the BBC computer requires a nominal input range of 0 to 1.8 volts, and VR1 is adjusted to bring the output voltage of the circuit somewhere near the middle of this range under normal operating conditions. Note that IC3 is a CA3140E device which has an output stage that enables output voltages of only marginally more than the negative supply potential to be achieved. It will also work well from a 5 volt supply. Most other operational amplifiers (such as the 741C or LF351) will not work well at this supply voltage and will not give a low enough output voltage for use in this application.

Diodes D1 to D3 effectively form a low voltage zener diode which prevents the output potential of IC3 exceeding more than about 2 volts so that only a marginal and insignificant overload of the computer's analogue port can occur. The three capacitors are all decoupling components. Like the heart rate interface, this one is powered from the 5 volt supply of the computer. A reasonably stable supply is required, but the 5 volt output of the BBC machine seems to be perfectly adequate in this respect.

The circuit is very sensitive, and a change of only about 10 or so degrees Centigrade is sufficient to drive the output over the full 0 to 1.8 volt range.

Construction

Both the interfaces are accommodated on a single printed circuit board, as shown in **Fig 3.** The two circuits are entirely separate, and the board has purposely been designed so that it is an easy matter to build just one or other of the circuits if desired. The top section of the board is for the heart rate monitor while the lower section is for the temperature interface.

Construction of the board is quite straightforward, but the CA3140E used for IC2 and IC3 has a MOS input stage. These should be fitted in 8 pin DIL sockets and the normal MOS anti-static handling precautions should be observed. The CA3140T offered by some suppliers is suitable for use in the IC2 and IC3 positions, but this version of the device

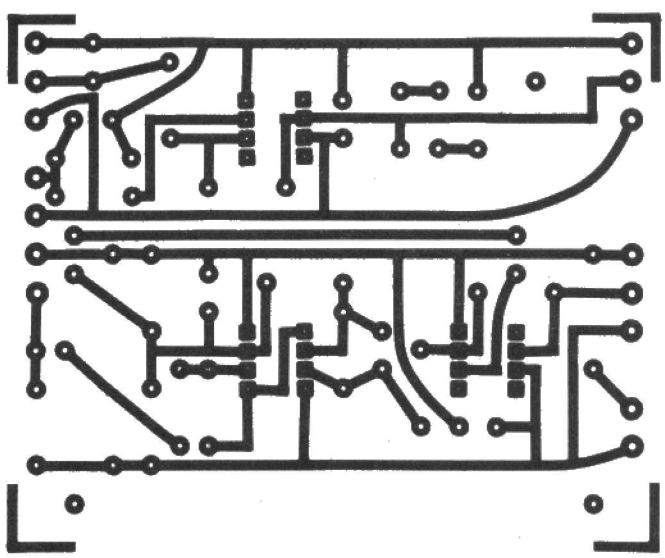


Figure 3. Foil pattern for both the Heart Rate and temperature measuring circuits. The board has been designed to allow just one of the circuits to be built if desired.

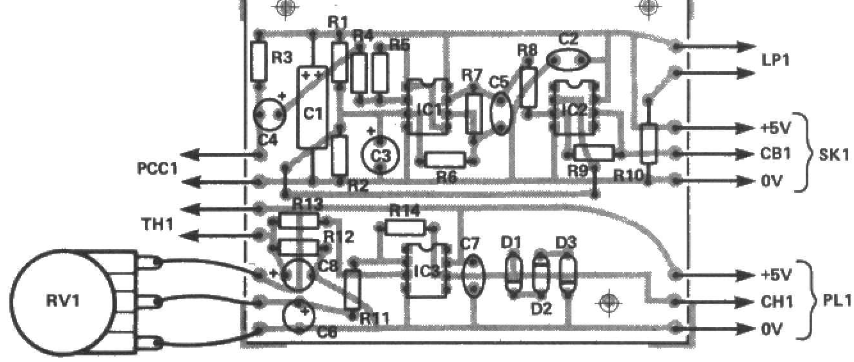


Figure 4. Overlay of the two lie detector circuits.

has a different case style and is likely to cost about twice as much.

Fit Veropins to the board at points where connections to off-board components will be made, and then generously tin these with solder.

On the prototype the connections to the two sensors are made via two 3.5mm insulated jack sockets (insulated types being used to avoid unwanted connections between the sockets through the metal front panel). Each sensor is fitted with a screened lead about 1 metre long and terminated in a 3.5mm jack socket. Use tape or sleeving to insulate the leads of the thermistor and photocell, plus the connections to these. This prevents the leads from accidentally short-circuiting together and also avoids any electrical connection between the circuit and the user. There is actually very little risk of an electric shock being received from the computer as it adheres to rigid safety standards, but an additional set of insulation between the user and the mains obviously does no harm. It also prevents electrical noise picked up in the users body from being coupled into the interfaces.

The connections to the computer are taken via two three way leads which do not need to be screened types. On the prototype these are wired direct to the printed circuit board, but they could connect to the board via 3 way plugs and sockets (mounted on the rear panel) if preferred. At the computer end a 15 way D plug is required for the connections to the analogue port, and a 20 way ID connector is used for the connection the user port. The diagram in the BBC manual which show the connections to these two ports can be a little confusing, but **Fig 5** shows the correct method of connection for both of them. These are as seen looking onto the connectors of the computer.

Tł 'ightbulb and photocell could be built into a single assembly, with a hole into which the users finger tip would be placed. Although this method has the advantage of convenience in use, it is a little difficult to build a suitable assembly of this type, and this method can result in the circulation to the finger-tip being hampered unless great care is taken. A simple alternative is to mount LP1 on the base panel of the case in a batten lampholder, and to drill a hole about 12mm in diameter directly above it in the top panel. With the photocell taped to the fingernail of the users right index finger (the left one for someone who is left handed), the fingertip is placed over the cutout in the top of the case.

Movement will produce spurious triggering of the unit and should be avoided. The photocell is more sensitive on the side to which the leadout wires do not connect, and results will probably be best with this side facing downwards. Mains lighting is modulated with the 50Hz mains frequency and this could also produce spurious triggering of the circuit. If necessary use several layers of tape over the photocell to keep the mains lighting at bay.

The thermistor is taped at any convenient place on the same finger as the photocell. It will take the photocell a while to adjust to your skin temperature which is likely to be significantly higher than the room temperature. VR1 is adjusted so that a reading of around 32000 (i.e. about half the maximum value) is returned from analogue channel 1 of the computer at the start. The suggested software includes a simple setting up program.

In use the basic idea is to sit back and try to relax whilst monitoring results on the TV screen. If this indicates that you are successful, then you continue in the same vein. If not, then a different tack can be tried. The equipment will show the effect of things such as listening to quiet music, or watching an exciting sporting event. Gradually the biofeedback should enable you to gain, by empirical means, greater control of the functions that govern stress and relaxation.

An innocuous way of trying out the lie detector aspect of the interface is to ask a subject to choose one card from a number of playing cards. Once the system has settled down they can be shown each card in turn and asked to respond with the phrase 'no that is not the card I chose'. When they 'lie' it should be possible to see a rise in heart rate and a corresponding drop in the temperature reading.

Program Notes

The software displays a graph on the screen of either the user's heart rate or skin temperature. The choice is made at the beginning of the program, which includes instructions. The program also guides the user through the initial setting up.

In heart rate mode a new section of graph is drawn each time a heart beat is detected, and a square flashes in the corner of the screen. The screen is automatically cleared and redrawn when the trace reaches the right hand side.

The temperature mode is essentially similar to the heart rate mode, except that readings are taken at one second intervals, and there is no flashing square.

In both modes, once the drawing part of the program is entered it continues indefinitely. Pressing 'ESCAPE' RUNs the program from the beginning. Return to command mode is by pressing BREAK only. Line 40 should not be entered until the program is running successfully and all typing errors have been found.

Line 80 defines a user-character as a square, which is used in the heart beat display. An array is declared at line 90, and this is used to calculate the average time of the last five heat beats. This is used in glitch handling. The assembly language section is used to detect transitions on CB1 of the user port. Lines 130 to 150 perform the initial setting-up while lines 160 to 190 form a loop which repeats until a transition is detected. When this occurs lines 200 and 210 perform the necessary reset, and control returns to BASIC. Bit 4 of the peripheral control register at &FE6C is set to 1 so that positive transitions of CB1 set bit 4 of the interrupt flag register at &FE6D (although in this application it does not really matter whether the active transition of CB1 is positive or negative). Writing 1 to bit 4 of the interrupt flag register resets it to zero.

The rest of the main program calls the appropriate procedures which give instructions, draw the graphs, etc. PROCav and FNav are used to average the last five beat times, and this value is used if a glitch occurs and an unrealistic time is returned. PROCset-temp simply prints the current ADVAL(1) reading on the screen so that the "set" control of the interface can be adjusted to give a suitable reading. PROC-set-beat prints a square in the centre of the screen, then calls the assembly routine. When a transition is detected the square is rubbed out, and the program the loops back and prints it again. Once C4 has achieved the correct charge, everything is set up correctly, and a fairly steady rate is attained (ones heart rate normally fluctuates slightly), pressing any key ends the procedure.

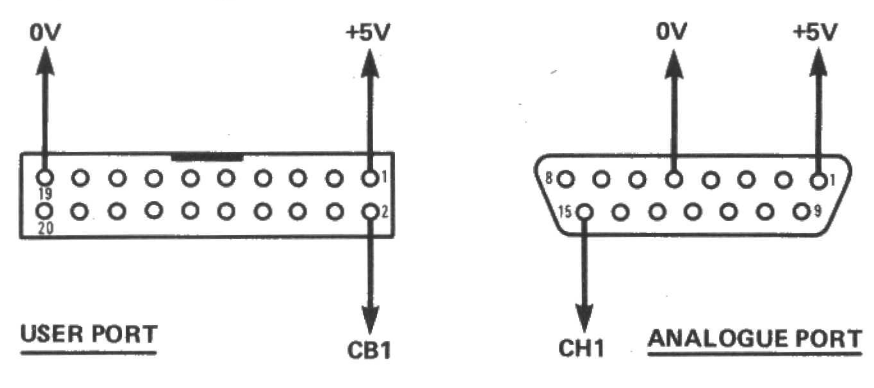
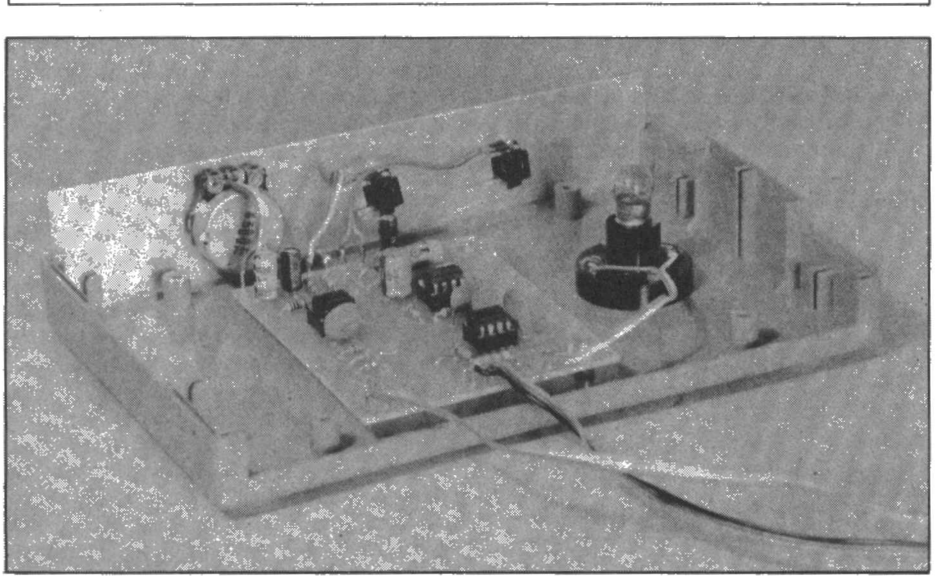


Figure 5. Connection details for both the user and analogue ports are shown as if looking onto the computer's connections.

```
10 REM BIOFEEDBACK
20 REM Program by John W. Penfold.
30 REM <C> AUGUST 1983
40 ON ERROR RUN
50 REM
60 MODE 6
70 VDU 20
80VDU 23,225,255,255,255,255,255,255,255,255
90 DIM avstore(4):FOR fill=0 TO 4:avstore(fill)=75:NEXT
100 DIM P% 50
110[ OPT 2
120.SET
130 LDA #16
140 STA &FE6C
150 RTS
160.START
170 LDA &FE6D
180 CMP #16
190 BNE START
200 LDA #16
210 STA &FE6D
220 RTS ]
230 MODE 6:PROCinstructions
240 MODE 1
250 IF choice$="H" OR choice$="h" THEN PROCbeat
260 IF choice$="T" OR choice$="t" THEN PROCtemp
290 END
300
990
1000 DEF PROCinstructions
1005 VDU 23,1,0;0;0;0;
1010 PRINTTAB(13,5);"BIOFEEDBACK"
1020 PRINTTAB(3,10);"This  Program  allows you to monitor"'"either  your  heart
rate  or  your  skin"'"temperature.  You must choose which now."
1030 PRINTTAB(5,20)"PLEASE ENTER CHOICE (H/T)"
1040 REPEAT
1050 choice$=GET$
1060 UNTIL choice$="T" OR choice$="t" OR choice$="H" OR choice$="h"
1070 CLS
1080 PRINTTAB(3,10);"You must now  set up the Biofeedback"'"Interface according
to the instructions"'"which follow."
1090 REPEAT UNTIL FNspace
1100 IF choice$="H" OR choice$="h" PROCset_beat
1110 IF choice$="T" OR choice$="T" PROCset_temp
1120 ENDPROC
1130
1990
2000 DEF PROCscreen
2010 VDU23,1,0;0;0;0;
2020 CLS X%=100
2030 GCOL 0,2
2040 MOVE 100,900
2050 DRAW 100,300
2060 DRAW 1200,300
2070 MOVE 100,600
2080 IF choice$="T" OR choice$="t" COLOUR 1:PRINTTAB(10,28);"TEMPERATURE MONITOR"
2090 IF choice$="H" OR choice$="h" VDU 19,3,4,0,0,0:PRINTTAB(10,28);"HEARTRATE MONITOR"
2100 ENDPROC
2110
2990
3000 DEF PROCdraw(Y)
3010 X%=X%+4
3020 GCOL 0,1:DRAW X%,Y
3030 IF X%=1204 PROCscreen
3040 ENDPROC
3050
3990
4000 DEF PROCbeat
4005 PROCscreen
4010 TIME=0:CALLSTART
4020 T=TIME
4030 IF (T<30) OR (T>120) T=FNav
4040 PRINTTAB(1,1);CHR$(225)
4050 PROCdraw(900-(5*T))
4060 PROCav(T)
4070 PRINTTAB(1,1);CHR$(32)
4080 GOTO 4010
4090 ENDPROC
4100
4990
5000 DEF PROCtemp
5005 PROCscreen
5010 TIME=0
5020 REPEAT UNTIL TIME>100
5030 PROCdraw(FNtemp(ADVAL(1)))
5040 GOTO 5010
5050 ENDPROC
5060
5990
6000 DEF PROCav(time)
6010 FOR C=4 TO 1 STEP-1
6020 avstore(C)=avstore(C-1)
6030 NEXT C
6040 avstore(0)=time
6050 ENDPROC
6060
6990
7000 DEF FNav
7010 total=0
7020 FOR C=0 TO 4
7030 total=total+avstore(C)
7040 NEXT C
7050 =total/5
7060
7990
8000 DEF FNtemp(a)
8010 a=a-10000
8020 a=a/100
8030 =a+300
8040
8990
9000 DEF FNspace
9010 PRINTTAB(5,23);"PRESS <SPACE> TO CONTINUE"
9020 =GET=32
9030
9990
10000 DEF PROCset_temp
10010 CLS
10020 PRINTTAB(3,10);"Adjust the reading to 30000 approx."
10030 PRINTTAB(5,23);"Press any key to exit"
10040 REPEAT
10050 PRINTTAB(15,12);ADVAL(1)
10060 UNTIL INKEY$(20)<>""
10070 ENDPROC
10080
10990
11000 DEF PROCset_beat
11010 CLS
11020 PRINTTAB(3,10);"Wait until square flashes"
11030 REPEAT
11040 PRINTTAB(19,12);CHR$(225)
11050 CALLSTART
11060 PRINTTAB(19,12);CHR$(32)
11070 PRINTTAB(5,23)"Press any key to exit"
11080 UNTIL INKEY$(20)<>""
11090 ENDPROC
```

Full program listing of the software to convert the outputs from the two circuit blocks into a graphical representation of heart rate and body temperature on the BBC micro's screen.



An internal view of the completed lie detector interface.

PARTS LIST

Resistors, 0.25W 5% except where noted

R1, 2	3k9
R3	2k2
R4	100k
R5, 7	1M
R6	1k
R8, 11, 13	10k
R9	47k
R10	3R3 0.75W
R12	1k5
R14	33k

Potentiometer

VR1	10k lin carbon

Capacitors

C1	100u 10V
C2, 7	100n ceramic
C3	220u 10V
C4	2u2 63V
C5	47n ceramic
C6, 8	10u 25V

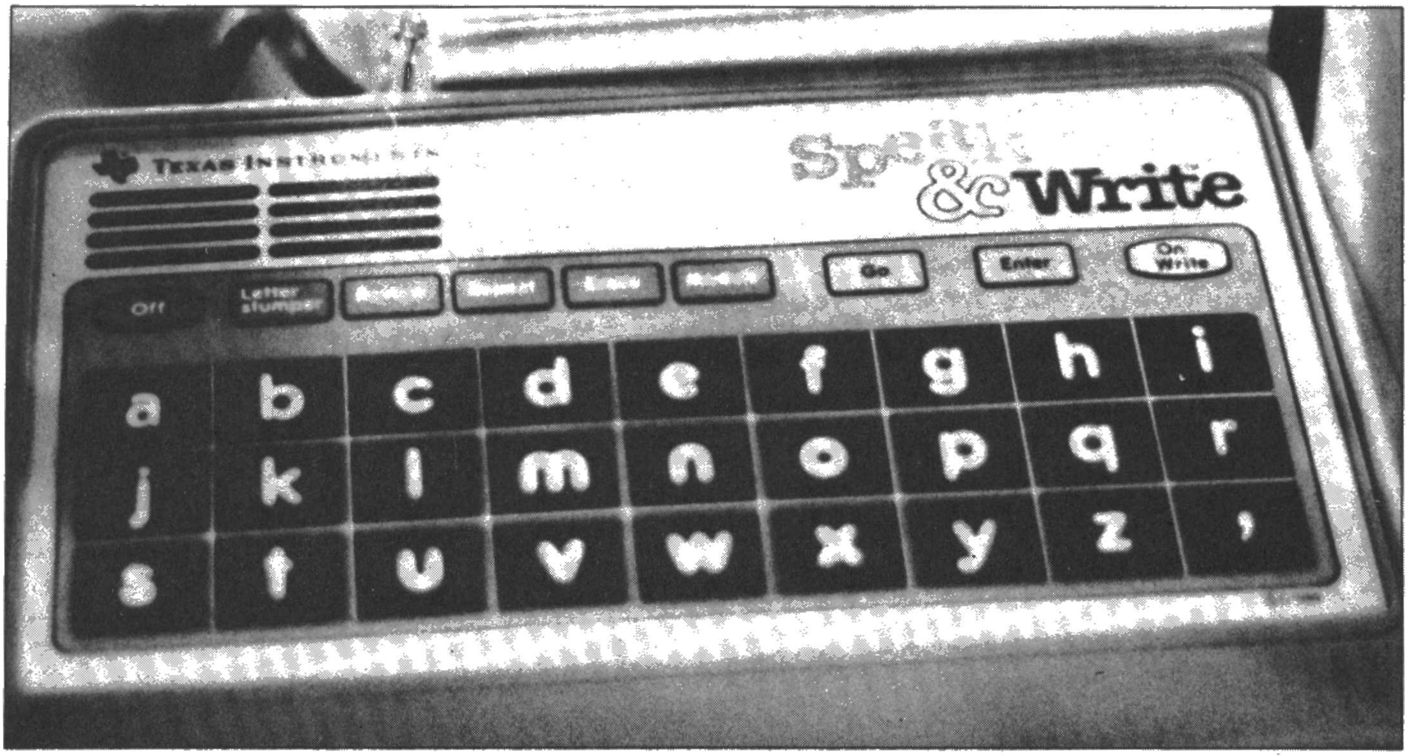
Semiconductors

IC1	1458C
IC2, 3	CA3140E
D1, 2, 3	1N4148

Miscellaneous

PL1	15 way D connector
SK1	20 way IDC header socket
LP1	3.5V 0.3A MES type
PCC1	RPY58A
Th1	VA1066s

Case, PCB, DIL sockets, MES batten lampholder, Control knob, Wire, cables, Veropins, fixings, etc. **E&CM**



DIGITAL SPEECH SYSTEMS

Ian Campbell eavesdrops on the latest in talking chips.

We must firstly ask why has digital speech been developed at all. The answer is really pretty obvious if you think about it. Man has had a fascination for countless ages with machines of some sort that interact with him by talking back when instructed to do so in some way.

Even up to recent times talking machines have been devices based on mechanical systems. These have been very limited in their ability to communicate information (cf British Telecom's talking clock). In addition they have been costly to install, costly to keep working and very unreliable.

What has been needed is a simple electronic system of producing good quality speech which could not only be mass produced but cheap as well. The market for such a device would be absolutely enormous and would expand dramatically once people realised it potential.

We are now witnessing such an expansion with the advent of workable electronic speech systems from a large number of the world's major electronics companies.

It must be put on record that the first battle honours in the field, for surely it is a battle to capture a very lucrative market, goes to Texas Instruments who marketed a toy in 1978 called "Speak and Spell". The electronics industries of the world were taken somewhat by surprise by the event. This was because although it was possible to produce electronic speech it was at the cost of thousands of pounds. TI had stolen a world lead by producing a very cheap system using a single IC based on LSI semiconductor technology and costing about 50 Dollars.

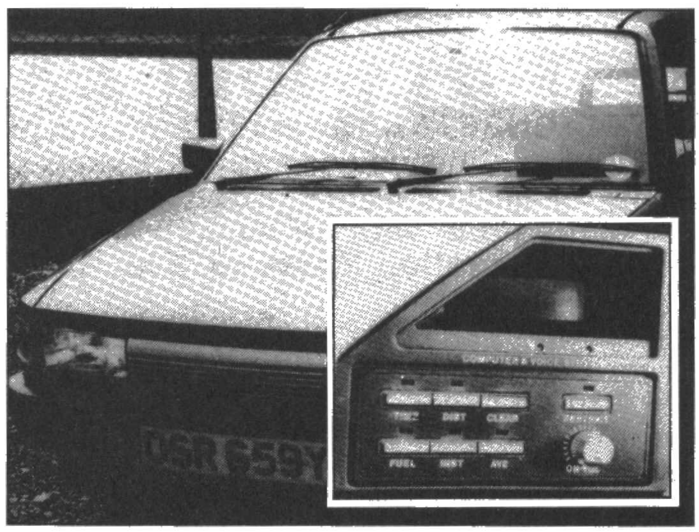
Since the introduction of TI's 'Speak and Spell', talking electronic devices seem to have taken rather a long time to come into the public's gaze. Speech synthesis has now hit the headlines in a big way, with the Prime Minister no less, driving a talking car backwards along Downing Street and many companies offering speech synthesisers for their respective computers not to mention a talking sewing machine and a space satellite with a 'Digitalker'.

In the next few years we might expect electronic speech synthesisers to become common place in industrial applications where there are probably almost unlimited possibilities for man-machine interfacing. As a way of illustration, at this time, if something on a complex bit of machinery requires the attention of the operator, a bell might ring or a light flash. A spoken warning would not only call the attention of the operator but at the same time tell him or her what action to take to correct the fault. This can then be done quickly, efficiently and without causing the blood pressure to rise.

Say a new piece of rather complicated equipment has been acquired. In industrial or consumer situations there exists the attraction of the equipment itself teaching the operator how to use it and at the same time telling him how well he is getting on.

In the purely consumer area, domestic appliances, bank cash dispensers, lifts, toys, watches, electronic games and learning aids will benefit (in some cases already are) from verbal instructions, warnings and encouragement.

After that rather weighty introduction this is probably a good time to have a look at just hom common or garden human speech is generated before venturing into the world of electronics.



Speech synthesisers are finding their way into all manner of products from sewing machines to cars.

Human Speech Synthesis

No one description of how electronic speech may be synthesised, for there are a number of ways, would be possible unless firstly we have a look at how the real thing is accomplished. It will then be possible to see how an electronic model can be constructed to emulate our vocal mechanism.

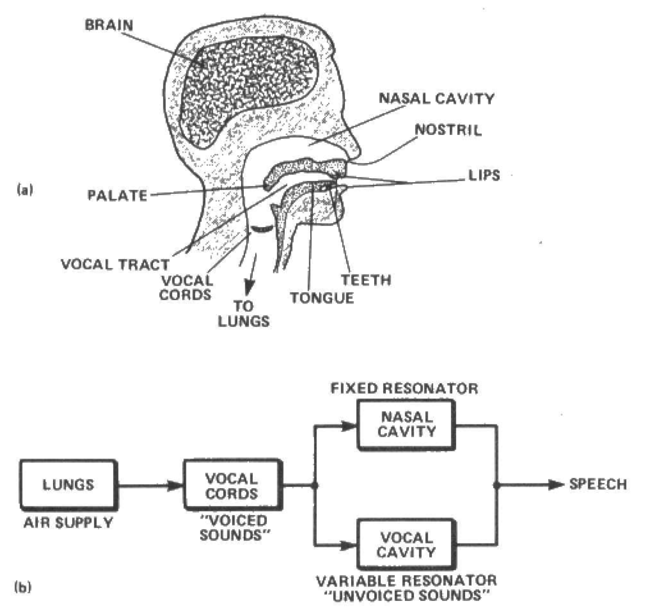


Figure 1(a). A simplified representation of the human vocal tract and (b) a block diagram of the system.

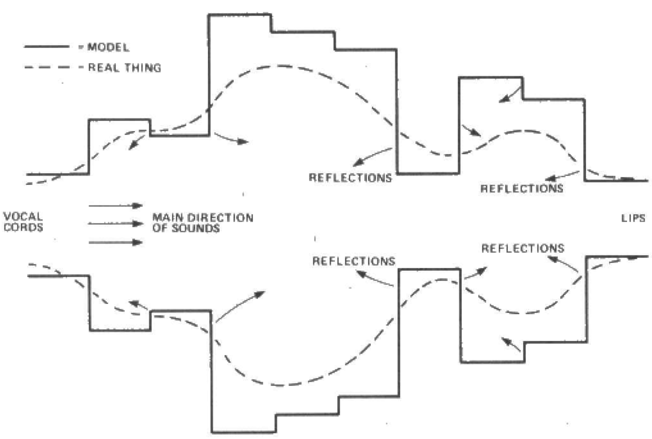


Figure 2. The human vocal tract may be modelled as a series of cylinders each with different diameters. Reflections occur where the different size cylinders 'connect'.

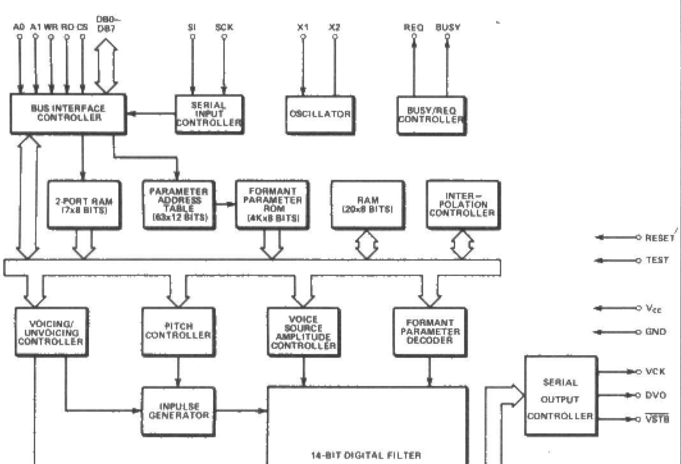


Figure 3. Block diagram of NEC's formant based synthesiser. This system does the majority of synthesising digitally with an A/D converter as the last stage in the process.

Figure 1 shows the human vocal tract and a simplified block diagram of the system. As can be seen, the main power source for our speech is the lungs. Air is forced out of these as the rib cage and diaphragm relax. This air travels up the trachae through the voice box, mouth and nasal passages. As the air passes through the vocal cords they vibrate. This produces a sound of a frequency determined by, the length of and tension in, the cords. The chambers formed by the mouth, nasal passages and throat resonate and add character to these vibrations thus forming what are known as voiced sounds as in 'a', 'e', 'i', 'o' and 'u'. Air just rushing through the vocal tract produces unvoiced sounds eg the 'sh' in 'shout' or the 'th' in 'that'.

It would now be an idea to look in just a little more detail at the basics of human speech formation. The simplest phonological element of speech is the phoneme. It is, if you like, a type of building block from which speech is constructed. A phoneme is a very simple sound that by itself cannot distinguish different words. Put a number of them together, add speaker inflection, volume, emphasis etc. and hey presto speech.

Since phonemes are so important in speech it would be an idea to say just a little more about them. Firstly there are about 40 different phonemes in English ie 14-16 vowel sounds and 24 consonant sounds. Each phoneme is made up of either a voiced sound or an unvoiced one. The spectral nature of the voiced sounds is very much determined by the physiological characteristics of the speaker's vocal tract. **Fig 2** shows a model of the tract and illustrates how each person can have a different voice pattern. The tract is treated as a series of interconnected cylinders with gradually varying diameters. The regions where the different size cylinders connect as it were, represent areas of impedence mismatch. At such junctions reflections of waves occur which causes resonance. Since the physical dimensions of everybody's vocal tract will be somewhat different, we will all have different resonances and therefore different voices. In addition we must not forget the contribution of the nasal resonance chamber which will also add its individual share to the final product (we have all heard of people who talk through their nose). The resonant chambers thus formed, act as filters and divide the sound up into a series of frequency bands called formants. A speaker, by controlling the physical nature of these chambers with mouth position, tongue position and throat orifice size, controls the formants which in turn generate a phoneme.

The unvoiced sounds do not in fact generate these frequency bands. They are characterised by very little resonance, if any, and more by noise or hiss. What we have here are in fact called fricative formants.

Electronic Speech

To the uninitiated the obvious way of obtaining electronic speech is to have someone talk into a microphone which is connected via an amplifier to an analogue to digital converter. The digitised audio could then be placed into a memory and withdrawn as and when needed. Unfortunately this method requires an absolutely staggering amount of memory space, even for short lengths of announcement time. Limiting the upper frequency limit of the recorded speech and then storing it as PCM or delta modulation signals in semiconductor memories still needs a massive amount of memory space (one second of digital speech requires between 16K and 100K bits of memory).

The problem therefore is one of reducing the memory requirements of the system to manageable proportions. The modern approach is to use the technique of speech synthesis. Three typical ways of doing the job are these:-

1. Playing back digitised, pre-recorded, basic units of speech, in the correct sequence to produce words.
2. The recall of speech waveforms that have been digitised then compressed – for example by Adaptive Delta Pulse Code Modulation.
3. The synthesis of speech using speech waveforms that have been analysed into several parameters and recorded digitally.

Speech Synthesis

In an earlier section it was said that formants were the generators of phonemes. These in their turn were, together with volume, emphasis etc., the fundamental building bricks of speech. It would thus seem logical that a formant based synthesiser could be constructed. The synthesiser would have to have an impulse source for the voiced sounds and a noise source (hiss noise) source for unvoiced sounds. In

addition the synthesiser would require a minimum of three formant filters for the voiced sounds and one for the fricative sounds. In this configuration for a male speaker the three voiced sound formants would occur in the bands 200Hz-800Hz, 900Hz-2300Hz and 2400Hz-3000Hz. The fricative formant would be high frequency noise in the band from 2500Hz-8000Hz. There would be a variation in amplitude between the 'th' sounds and the 'sh' sounds, with the latter being the loudest.

The minimum formant synthesis technique would require formant filter co-efficient data, amplitude control data and data to control the driving of the voiced impulse and unvoiced noise generators, which would all be stored as digital information in semiconductor ROM. Although the system would not give quality speech, the ROM would only need to supply data at an approximate rate of 400 bits per second of speech, which is a marked improvement over the D to A technique originally described.

NEC have gone the road of the formant based synthesiser, a block diagram of their single chip system is shown in **Fig 3.** As can be seen this is certainly not a minimal setup and as such requires a low data rate of approximately 2400 bits/sec. As many be concluded from the diagram, all the speech synthesising is done digitally right up to the last stage where there is an analogue to digital converter. This produces the final audio output. A quick flick through the circuit blocks will reveal a lot of familiar words like "format parameter ROM", "voice source amplitude controller" and "noise generator", etc. It is not the purpose of the article, however, to become deeply involved in the nitty gritty of what controls what and when in every type of speech synthesiser. We will therefore dwell no longer on this particular one, mainly because the diagram is pretty much self explanatory except maybe for one or two of the controlling sections.

General Instrument has taken a someqhat different path to that of NEC with their SPO256 synthesis chip called the "NARATOR". They have used allophones as the basic units to be concatenated together to synthesize speech.

About Allophones

A short word about allophones would be useful here before looking a little at the workings of a GI synthesiser. In an earlier part of the article the word phoneme was mentioned. A phoneme remember is the name given to the building blocks of speech ie the basic sounds which are the vowels and consonants. These basic sounds may be acoustically different depending on where they appear in a word. The (p) sound, for instance, is aspirated when in the initial position of "pop", unreleased in the final position and unaspirated in the word "spy" since it follows the (s) sound. These various possible (p)s are called allophones of the phoneme (p). An allophone is thus an acoustic variant of a phoneme. Since there are about 40 phonemes in English it means that there must be more allophones. In fact there are, for practical purposes, 64 recognised by GI for use in their synthesiser.

This small number of "units" from which speech can be built has a distinct advantage over earlier methods as regards memory size is concerned. Remember the name of the game is to have as low a bit rate as possible. Speech uses approximately 10-12 allophones per second which means a bit rate of no more than 100 per second when the allophone ROM needs a six-bit address (as does GI's). There is one big problem, however, with an allophone based synthesiser and that is the quality of the speech which it is capable of producing.

This problem happens because when a person is talking there is a continuously varying speech waveform produced. During this process co-articulation takes place. That is before one phoneme is finished another is started. As a result of this it is impossible to obtain a speech sound that is pure. An example of this is found in the word "back". If an attempt is made to isolate the (b) sound by taking successively longer samples of the sound waveforms starting at the beginning of the word, the first thing heard is noise, then a (ba) sound will be noted at a certain juncture. So at no time will the (b) sound be heard in isolation. That is why an allophone based construction of the word "back" will not sound quite right because it will be made up of a (b) sound followed by an (a) sound. It is for this reason that when words are constructed using allophone synthesis that great care has to be taken in order to ensure that they sound as natural as possible. In fact it is not as easy as it might at first appear.

The allophones used by GI and some guidelines as to how they sound and also where to use them were shown in last month's article on Cheetah's Sweet Talker. It makes rather interesting if not

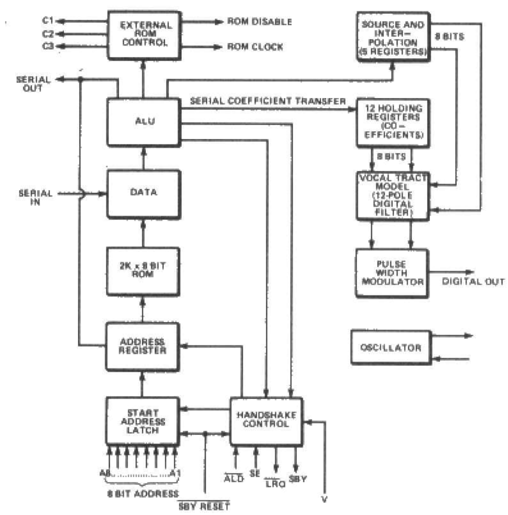


Figure 4. GI's SPO 0256 is to be found at the heart of many speech synthesis systems. It is capable of producing 64 allophones.
By concatenating combinations of these the system is capable of producing an unlimited vocabulary.

confusing reading. The main point about constructing words using the GI allophones is to forget about the spelling of the desired word and just think about how it actually sounds. This is quite important because many words contain hidden sounds which are not reflected in their spelling eg the word "uncle" is pronounced "ungkul" and the word "computer" has a hidden (y) sound.

The next step is to decide which allophones are most suitable to make up the word. The story does not end there since to make the speech more realistic, pauses will have to be inserted between words, even though natural speech has none. **Table 1** shows some examples of how allophones may be concatenated to form words.

TABLE 1

COMBINING ALLOPHONES TO MAKE SOME WORDS

DD2-A0-PA3-TT2-ER1	daughter
KK3-AX-LL-AY-PA2-DD1	collide
KK1-AX-MM-PA3-PP-YY-UW1-PA3-TT2-ER1	computer
SS-SS-IH-SS-PA3-TT2-ER1	sister
AX-NG-PA3-KK3-EL	uncle

The SPO 0256 block diagram shown in **Fig 4** contains all the necessary bits and pieces, when used with a few support components, to make a simple speech synthesiser. Its mode of operation is basically fairly easy to understand. The main controlling section is the ALU or microcontroller. It controls the flow of data from the ROM (the ROM contains the 64 allophones and a program of instructions) to the 12-pole digital filter. This data will contain the necessary instructions to make the filter model the voal tract to suit the allophone being synthesised. In addition, the ALU controls the assembly of the allophones. Its task does not quite end there because it also controls the amplitude and pitch information that is fed from the source and interpolation registers to the digital filter.

The story is finalised when the pulse width modulator, driven by the digital filter, produces a digital output which is then filtered by an external 5kHz low pass filter to produce the audio signal in a loudspeaker. Under normal operation a microcomputer would communicate the addresses of the allophones to the chip via the input address lines and controlling pulses would occur between micro and synthesiser via the handshake inputs.

The "Narator" chip is flexible enough to support external speech ROMs that are communicated with by the ALU using a serial data stream. It is also capable of supporting other types of synthesis systems like formant based and LPC of which more anon.

It was said that an allophone based synthesiser did not produce speech of super-high quality, there is, however, no loss of intelligability. The method does have the big advantage of being capable of producing any word. This lends itself to equipment turning written text, like this, directly to speech.

CONTINUED NEXT MONTH **E&CM**

SINGLE CHIP MICROCONTROLLER

Part 2

Last month Richard Whitlock introduced Motorola's series of self programming micro controllers. This month he moves on with details of the systems hardware together with some more on the MCU itself.

Last month the circuit diagram of a basic 6805 programming device was described together with a detailed look at its operation. **Fig 2** shows the component overlay of the programming board while **Fig 3** shows the board's foil pattern.

The four switches shown in last month's circuit diagram are all mounted off board and **Fig 1** shows the way in which the switches are connected to the main PCB.

Construction of the board is reasonably straightforward although when finished it pays to ensure that all components are correctly inserted and, in particular, that all the wire links are inserted.

System Operation

With the details presented both here and in last month's article, those of you keen to go it alone should have enough information to start programming and using the device in some basic applications. We'll be presenting some applications for the system in months to come but will round off this article with some more details about the facilities of the MCU itself.

The Mask Option Register

The Mask Option Register (MOR) contains seven bits which have significance for the operation of the 68705, bit 3 being the only "don't care" bit from the programming point of view. The information carried by the various bits of the MOR is used during the micro-programmed sequence of operations executed by the chips' internal logic during the time that the internal Reset signal is active. Their effects are explained briefly below.

Bit 7 (CLK) – This bit, when programmed ot logic "1", causes the master oscillator circuit, from whose output all on-chip timing signals are derived, to be configured to accept a range of low cost external components as the frequency determining elements in its operation. When set to logic "0", the CLK bit causes the master oscillator to be configured to accept either a crystal timing element or an external clock signal. See the System Clock section below for further details of these two groups of modes.

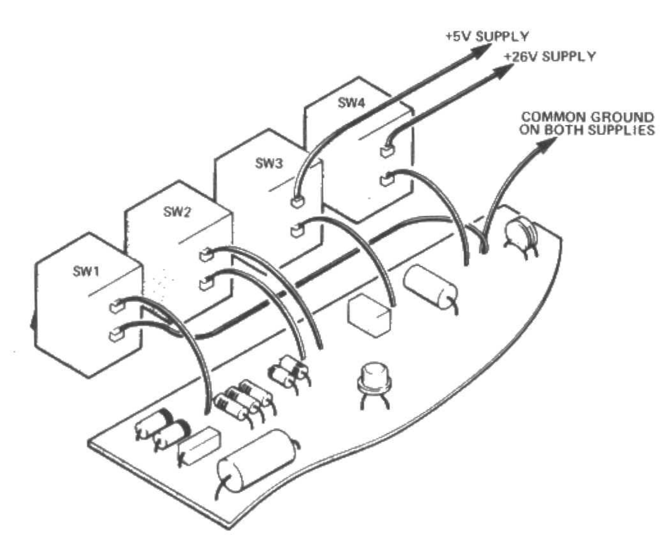


Figure 1. Connection details of the four off board switches shown in last month's circuit diagram.

Bit 6 (TOPT) – This bit, when programmed to logic "1", causes the on-board Timer to be configured in the same way as are the timers of the mask programmed 6805 family members, ie a single fixed Timer Clock source and a fixed Prescaler Ratio. In this case bits 5,2,1 and 0 of the MOR alone determine the operating mode of the timer by irrevocably configuring bits 5,4,2,1 and 0 of the timer Control Register, their logic values being directly transferred to the corresponding bit of the Timer Control Register (TCR) and bit 4 or the TCR being set to logic "1". In this case bit 4 of the MOR is also a "don't care" bit.

When TOPT is programmed to logic "0", the Timer is under full software control with bits 5,4,2,1 and 0 of the MOR merely setting the initial logic values of the corresponding bits in the TCR.

Bit 5 (CLS) – This bit selects between two possible clock signal sources for the Timer. When CLS is at logic "1", the operative clock signal source is the TIMER pin of the 68705 IC, and when CLS is at logic "0" the operative source is the processor clock running at a quarter of the master oscillator frequency.

Bit 4 – This bit is a "don't care" bit if TOPT is at logic "1", since the corresponding bit in the TCR is set to logic "1" in any case. When TOPT is at logic "0", bit 4 of the MOR is used to control the initial logic level of the corresponding bit in the TCR.

Bit 3 – A "don't care" bit in all cases.

Bits 2-0 (P2-P0) – These bits are used to specify, either permanently or initially, the Prescaler Ratio to be employed in Timer operations. See the Timer section below for the ratios decoded from these three bits logic values.

The complexity of the programming of the MOR stems from the need to provide a means of simulating all the mask option possibilities on an EPROM based system that is inherently more flexible than the system that it is modelling in the prototyping role. It is symptomatic of Motorola's recognition of the usefulness of the 68705s, apart from as prototypes for the related mask-programmed parts, that the user is given the software controlled timer mode as a bonus, when slavish following of the mask programmed parts' capabilities is the general rule amongst other manufacturers.

The Timer

The 68705 Timer can be more accurately described as a counter/timer since it includes logic elements designed to facilitate the performance of both functions with a minimum of processor intervention. As shown in **Fig 4** it consists of signal selection and gating logic followed by a 7-bit prescaler with eight stage outputs including a "stage zero" or bypass output, leading to the main 8-bit down counter which is directly accessible to the processor.

The initial signal, fed to the main down counter via the prescaler or the bypass, can be an external signal applied to the TIMER pin of the 68705 IC, the processor clock or the processor clock gated with an external signal. The first of these modes allows event counting to be easily accomplished, the second provides simple delay timing and periodic interrupt generation for functions such as data logging or time-of-day clock updating, while the third allows such functions as pulse-width measurement to be undertaken with great accuracy.

To the 68705 processor, the Timer appears as two registers, the main 8-bit down counter itself accessed through the Timer Data Register at address 008H (TDR), and the Timer Control Register (TCR) at address 009H. The TDR is a convenient read/write location which may be accessed for a read or write operation at any time without affecting the rate of counting. At Reset the TDR is initialised to FFH and, in the mask programmed simulation modes,

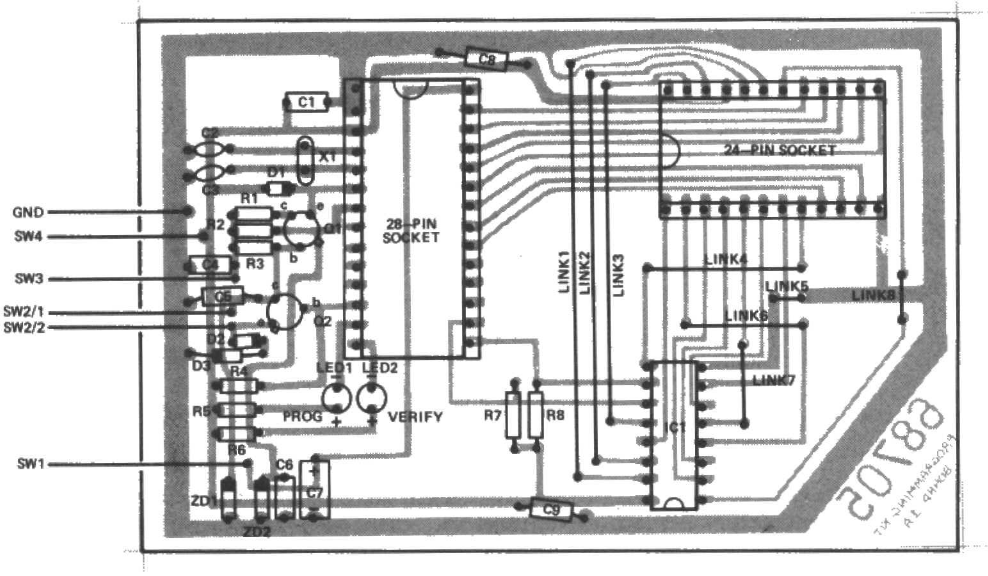


Figure 2. The overlay of the 68705 programming board. Note that because of the compact layout of the board some components may have to be mounted end on. The 68705 device is available both from Magenta and Technomatic both of whom advertise in this issue.

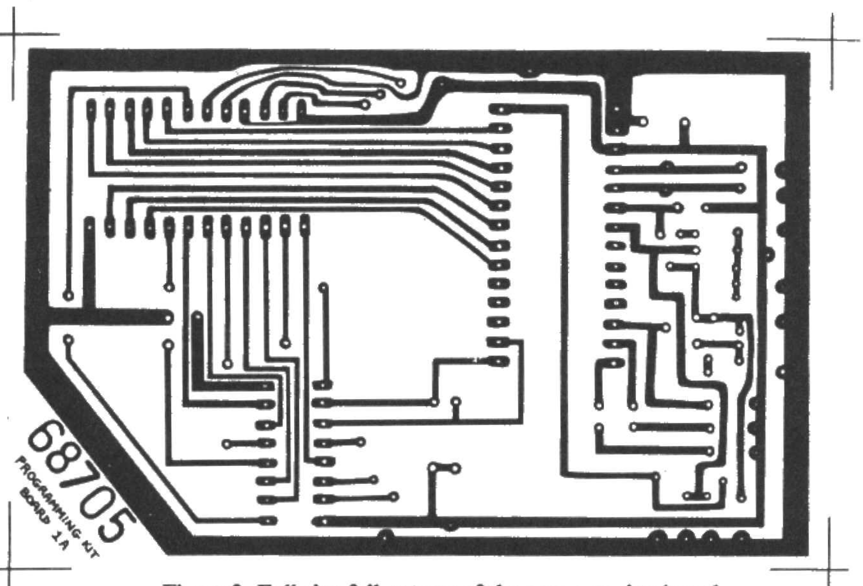


Figure 3. Full size foil pattern of the programming board.

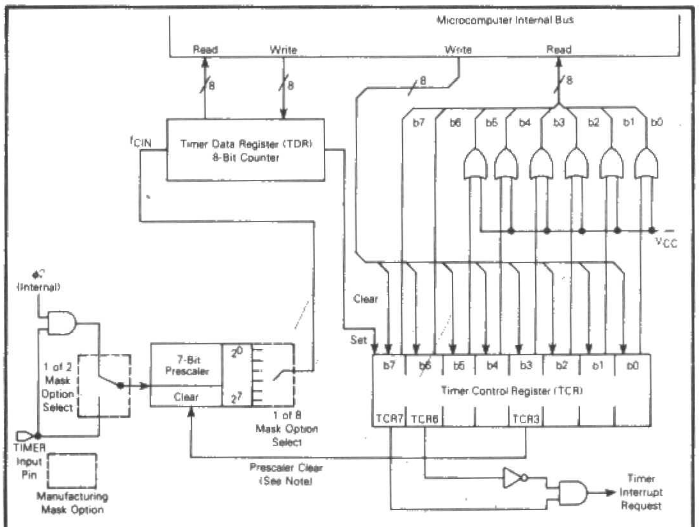


Figure 4. The major elements of the 68705's timer.

will begin to decrement, at the rate determined by the configuration set up in the MOR, immediately upon the internal Reset signal becoming inactive. However, if the software controlled mode is selected, it is possible to initialise the input circuitry so as to inhibit counting until the processor writes a new mode instruction to the TCR, by the right choice of MOR bit values, (see **Table 1**).

Not all the bits of the TCR are straightforward read/write bits. The functions and characteristics of each bit are given below.

TABLE 1

Summary of Timer Clock Source Options

TCR5	TCR4	Option
0	0	Internal Clock to Timer
0	1	AND of Internal Clock and TIMER Pin to Timer
1	0	Inputs to Timer Disabled
1	1	TIMER Pin to Timer

Bit 7 (TIR) – The Timer Interrupt Request Bit. This bit is cleared at Reset. It is set only by the main timer down counter decrementing through zero and, during normal device operation, can only be cleared by a processor write operation, with bit 7 of the written data at logic "0". When the timer interrupt is enabled the setting of the TIR

bit causes an interrupt request to be passed to the processor, where, if the main Interrupt Mask bit (I in the CC Register) is clear, a processor interrupt sequence is initiated. If either the local timer interrupt mask bit or the main Interrupt Mask bit is set, no interrupt request is received by the processor. However in this case the TIR bit can still be read under program control to see if a Timer Interrupt is pending. Note that the interrupt handling routine for a Timer interrupt must include an instruction to clear the TIR bit so that further Timer interrupts may be recognised, as clearing of the TIR bit is not automatic.

Bit 6 (TIM) – the Timer Interrupt Mask bit. Set on Reset, it may be both cleared and set by a processor write operation. When set it stops a Timer interrupt from being passed to the processor, but has no effect upon interrupts from any other source.

BIT 5 (TIN) – When the mask option emulation mode of timer operation is selected (see MOR section above), this bit is loaded once and for all time at Reset from the corresponding bit of the MOR and is then not accessible to the processor. In the software controlled timer mode the TIN bit is initially loaded from bit 5 of the MOR and is subsequently accessible to the processor for both read and write operations. When set, the TIN bit selects an external signal applied to the TIMER pin of the 68705 IC as the clock source for the Prescaler/Timer. When clear, the TIN bit selects the processor clock as the clock source for the Prescaler/Timer. The effects of TIN are modified by the state of Bit 4 (TIE).

BIT 4 (TIE) – When TIE is set an external signal applied to the TIMER pin of the IC is allowed either to drive the Prescaler/Timer directly (TIN "1") or to gate the processor clock signal before it reaches the Prescaler//Timer (TIN = "0"). When the TIE bit is clear, an external signal on the TIMER pin has no effect upon internal circuitry. The TIE bit is permanently set in the mask option emulation mode of Timer operation, and is not accessible to the processor. In the software programmable Timer mode the TIE bit is initially loaded from the corresponding bit of the MOR and subsequently may be set of cleared by a processor write operation. (For TIN – TIE interactions, see Table).

BIT 3 (PSC) – This is a write only bit which always reads as logic "0". Writing a logic "1" into this bit generates a pulse which sets the Prescaler to the all "ls" state. This enables the system to start a count from a known datum and avoids truncation of the first Prescaler cycle. The Prescaler is automatically set to the all "ls" state at Reset, so that a count may begin immediately, using the Timer mode set in the MOR. Wrioting logic "0" to the PSC bit has no effect. When changing the Prescaler division ration selected, (see bits 2 – 0, below), a logic "1" should be written to the PSC bit IN THE SAME WRITE CYCLE, to avoid possible spurious clocking of the main down counter.

BITS 2 – 0 (PS2 – PS0) – These three Prescaler stage select bitsd are decoded to select which, of the seven Prescaler stages and the bypass path, is to be the clocking signal source for the main down counter. **Table 2** shows the prescaling ratios selected by the different possible combinations of logic levels stored in these three bits. In the mask option emulation mode of operation, these three bits are loaded at Reset from the corresponding three bits in the MOR and are not subsequently accessible to the processor. In the software controlled Timer mode these bits are initially loaded from the MOR and then may be written and read by the processor under program control.

Prescaler

TCR2	TCR1	TCR0	Result	TCR2	TCR1	TCR0	Result
0	0	0	÷1	1	0	0	÷16
0	0	1	÷2	1	0	1	÷32
0	1	0	÷4	1	1	0	÷64
0	1	1	÷8	1	1	1	÷128

The TIMER input pin logic on all 68705s is TTL and CMOS compatable, but unlike the INT input pin discussed below it does not include Schmitt trigger circuitry to accommodate slowly varying signal levels, sharp logic level transitions are necessary. The author's experiments have shown that 20ns is the longest transition time tolerable for trouble free operation, though this will probably vary from one individual device to another and with temperature for a single device.

The Timer input pin has a secondary function. Circuitry connected to it detects a voltage of between 9V and 15V and, if such a voltage is present at the time that the internal Reset signal becomes inactive, causes the vector at addresses 7F6H and 7F7H for the 68705P3, or FF6H and FF7H for the 68705U3 and 68705R3, to be loaded into the Program Counter. These locations are the last two bytes of the mask programmed "bootstrap" ROM and hold the start address of the self programming routines. The detection of this high voltage at this time is only one of two conditions that must be met for the chip to enter the self programming mode. A high voltage must also be present on the Vpp pin, causing bit 2 of the Programming Control Register, at address 00BH, to be clear, before the processor is able to proceed with self-programming. (See the SELF-PROGRAMMING section below).

When a Timer interrupt has occurred, as the main down counter decremented through zero, it may be some considerable number of processor cycles before the interrupt is serviced. It may then be necessary to determine how long before service the interrupt occured. The main down counter continues to decrement at the prescribed prescaled rate and a read operation will recover a value equal to the complement of the number of prescaled Timer cycles that have elapsed since the interrupt was triggered. A COM instruction will produce the true value.

The maximum frequency, or minimum period, of a signal that can be recognised by the logic connected to the external TIMER input is gioven by:-

$$\frac{1}{f_{max}} = P_{min} = 2t_{cyc} + 250\text{ns}$$

for a 50% duty cycle signal, where t_{cyc} is the period of the processor clock and is equal to four times the period of the master oscillator. For high or low duty cycle signals, the period of the shorter "half wave" should not be below t_{cyc} + 250ns for trouble free clocking.

Continued Next Month

PARTS LIST

Resistors

R1	100R
R2, R3, R4, R7, R8	4k7
R5, R6	470R

Capacitors

C1, C4, C8, C9	100n ceramic
C2	100p ceramic
C3	68p ceramic
C5, C6	10n ceramic
C7	1u0 tantalum bead; 35V

Crystal

X1	1MHz AT-Cut//resonance

Semiconductors

D1, D2, D3	IN4001:50V; 1A
ZD1	22V; 400mW
ZD2	12V; 400mW
LED1, LED2	Standard Red L.E.Ds
TR1, TR2	2N2222
IC1	4040 CMOS Counter

Switches

S1, S2, S3, S4 SPST Switches

24-pin DIL Socket, 28-pin DIL Socket

ZIF sockets ideal, but high-profile, open-entry adequate with care.

Miscellaneous

PCB, Connecting Wire, Box etc.

Power Supplies

+5V @ 500mA, +26V @ 50mA

Connected to a Common Ground Rail.

MPF1-PLUS

REVIEWED

Ken Alexander assesses the new, upgraded version of the MPF-1 microcomputer.

The Microprofessor MPF-1 Plus is a redesigned and, as the plus might suggest, an upgraded version of the MPF-1 reviewed in our August 1982 issue. The unit as supplied could, in brief, be described as a stand alone, Z80 based, machine code development/educational computer although a series of add-ons can bring the specification of the MPF-1 Plus up to a system capable of running BASIC and supporting a thermal printer driver, EPRCM programmer board, Speech synthesiser and sound generation units.

Before going any further with this look at the MPF-1 Plus it may be as well to clear up any confusion that the name Microprofessor may cause. The name Microprofessor is a general term used by the Multitech Industrial Corporation for all of its microcomputer products. Thus the name is applied to the MPF-1 Plus as well as to the MPF-4, a business machine that the company are at present developing. The name was also used for a computer that recently became the subject of court action amid allegations from the Apple Corporation that it infringed certain of their copyrights. The MPF-1 Plus however bears no relation to this product.

Hardware Overview

Superficially the new version looks very much the same as its predessor and many of the good features of the MPF-1 have been retained. For example the basic board is still supplied in a sturdy box file that serves both to protect it during transit and to form a convenient home for the computer when in use. Also evident is the high quality of the double sided PTH PCB on which the MPF-1 Plus is constructed.

The differences between the two units begin with the keyboard for although the well liked positive feel of the original design have been retained, the 36 keys of the MPF-1 have been replaced by 49 keys that provide a full alphanumeric keyboard in addition to various control keys – this in contrast to the 16 Hex keys and 19 control plus one user definable key of the older board.

The display has also been upgraded when compared to the MPF-1 and now allows display of a full 20 characters formed on 16 segment fluorescent indications. Again contrast this to the four plus two display of the MPF-1.

The PCB itself is dominated by three 40 pin ICs, one of these being the Z80 MPU (top left) while the others (just behind the display) are 8255 PIAs. These provide a very generous 48 I/O lines for two way communication with other devices.

The Z80 is driven at the same clock rate as the earlier Microprofessor, namely 1.75MHz and this is again derived from a 3.579MHz crystal controlled clock.

Next to the Z80 is an 8K 2764 EPROM that holds the system's monitor – a much improved version of the original but more on that later.

The systems RAM is provided by two TMM 2016Ps – static 2K x 8 bit devices – providing a total of 4K (twice as much as the original machine).

A useful facility provided by the MPF-1 Plus is the ability to back up data in RAM by a battery power supply consisting of four UM3 batteries housed under the board. With the NMOS RAMs fitted as standard data will be preserved for about five hours and, if CMOS HM6116 devices are fitted, data can be retained for anything up to a year. The battery back-up can be disabled by means of a slide switch along the left hand side of the board.

A spare socket between the EPROM and RAM devices allows an additional 2764 or 2732 to be fitted on board.

The small, 2¼" spacer on the original MPF-1 is retained although it has now moved to the rear edge of the PCB.

Connections to a tape recorder for data storage are again via two 3.5m jack sockets at the rear of the PCB. The cassette interface has been designed for reliability not speed and transfers data at the rate of 165 bits per second.

Just below the tape I/O sockets are two LEDs, the green device lighting to indicate that a key press has been detected (the speaker also emits a beep upon a key press), while the red LED lights to inform the user that the Z80 has been HALTED.

The MPF-1 Plus requires an unregulated 9V DC power supply (an on board 7805 proves a stable 5V line) and this is provided by a ubiquitous black plastic plug and combined transformer. This is an improvement on the original adaptor which, as noted in our August 82 review, was of an American two pin type.

Finally something that is missing from the new Microprofessor – the prototyping area. The previous machine had a small area of the PCB set asside for wire wrapping interface circuitry but this area will not be missed too much as it is almost certainly easier to develop circuitry on separate breadboards and to connect it to the MPF-1 Plus via the units 40 WAY expansion convector.

The Monitor

As mentioned above the monitor resides in an 8K EPROM and, as well as the more attractive facilities of the MPF-1 Plus, takes care of the routine housekeeping functions of the system.

Upon power up the firmware initialises the system, begins scanning the keyboard and displays the contents, if any, of the systems display buffer.

The monitor also takes care of the tape loading/saving system which, as discussed, operates at 165 BPS. The system incorporates a checksum error detection system and allows any file/data stored on tape to be given a filename.

The major monitor commands are shown in **Table 1** which shows that the MPF-1 Plus supports a wide range of commands that enable the data stored in memory and the Z80's registers to be displayed and altered.

The monitor also allows breakpoints to be set within a program and, although only one breakpoint is allowed, this is a powerful debugging aid providing the user with the opportunity to examine the contents of the MPU's registers at the breakpoint. Together with the STEP facility, allowing memory locations and registers to be examined after each instruction is executed, determining just what is happening within any program is made quite straightforward.

As with the original MPF, the new machine provides the handy facility of being able to calculate the displacement for relative jumps both forwards and backwards.

Assembler

A major improvement evident in the MPF-1 Plus is the inclusion of both one and two pass assemblers within the firmware of the system.

The one-pass assembler converts lines of assembly language into machine code upon entry and thus cannot handle any pseudo instructions and will not accept symbols and labels. The one-pass assembler must be provided with absolute addresses and displacements. The major advantage of this assembler is that it does not require any memory in operation.

The two-pass assembler can process pseudo instructions and during pass one will fetch an labels in a source program and will create a symbol table which contains the labels and their corresponding values. During the second pass the assembler uses these values to create the object code.

The assembly language conventions of the MPF-1 Plus are in the mian similar to Z80 standard commands there being three major exceptions, namely that a comma is used to separate operations, a semicolon must precede each comment and that the system only accepts values in base 10 or 16.

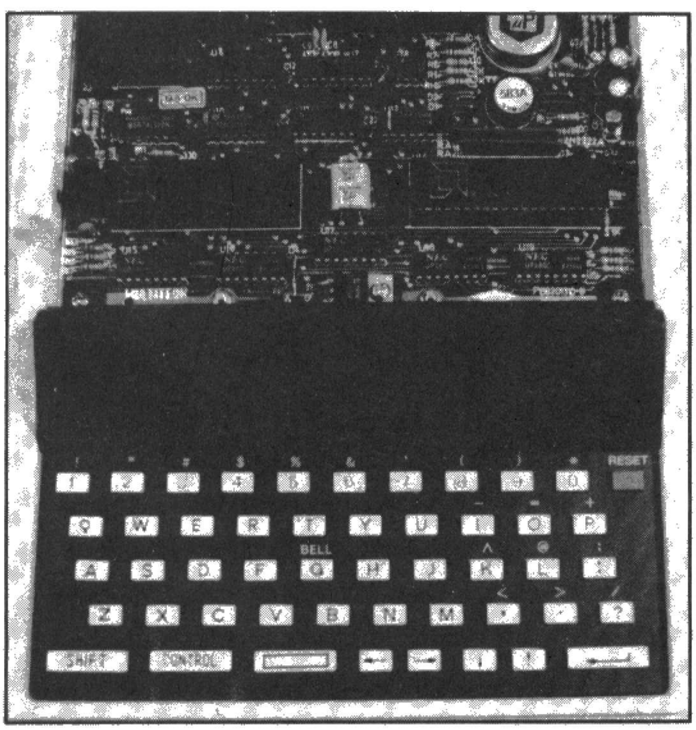
Manual Description

The standard of documentation provided by the Multitech Corporation to accompany the MPF-1 Plus sets a level that many other people could do well to copy. Three manuals are supplied, the fatest one being the users manual that describes the system in exhaustive detail, providing such useful information as device pin-outs, hardware configurations and full details of the Z80's instruction set. A second manual provides a fully annotated source listing of the system's monitor and together with the information provided in the main users manual, allows many of the monitor's routines to be incorporated within users program.

The third manual is a hardware/software experimenters guide and provides a range of routines covering items such as binary to BCD conversion and a square foot routine to things like a microcomputer organ and a music box simulation.

The three books together combine to provide a wealth of information that is presented in a clear and logical fashion.

A close up view of the alphanumeric keyboard. The high quality of the MPF-1P's PCB is evident from the photo.



Conclusion

The MPF-1 Plus is a distinct improvement on the MPF-1 with many powerful features added to the facilities of the original machine.

The standard of construction was high and coupled with the wealth of documentation provided, the system is indeed a powerful software development look.

The computer is a powerful stand alone machine and with the range of add-ons produced for the system can grow into a very flexible computer.

The add-ons include an EPROM programmer (available early next year), a 20 character thermal printer that also endows the MPF with a Z80 disassembler and speech and sound boards. Also available soon is a Z80 PIO and CTC board.

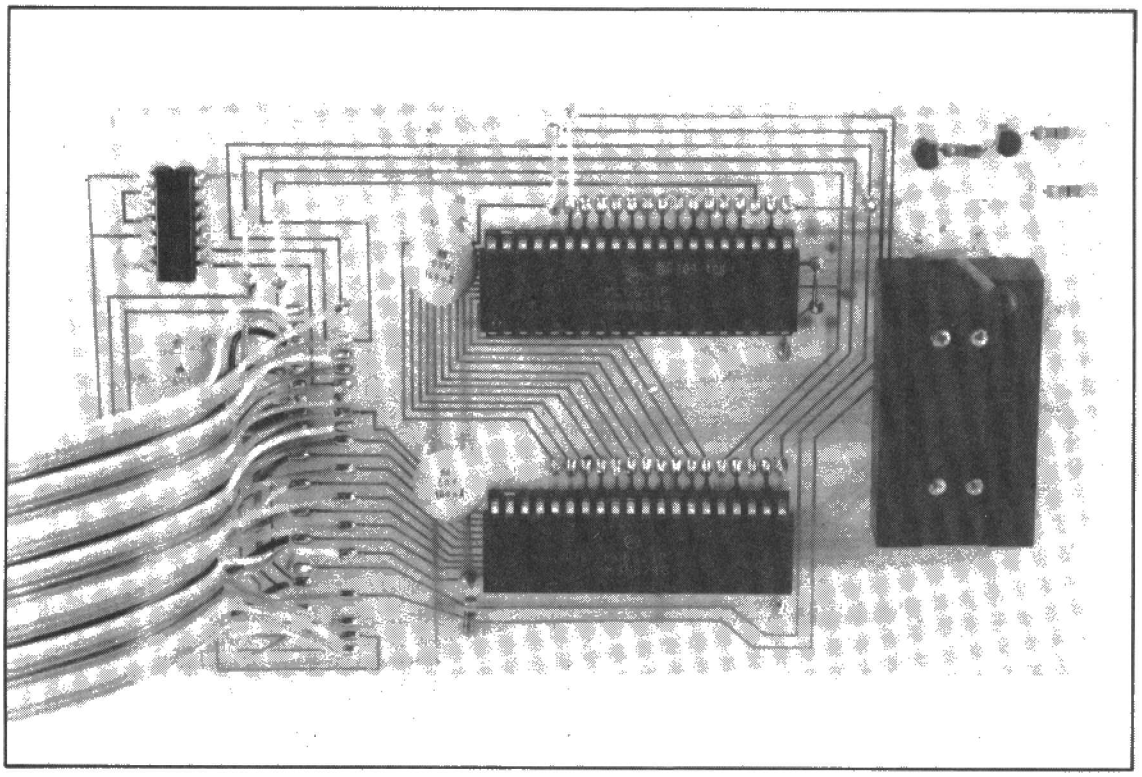
At £140 the plus version of the system is more expensive than many personal micros but this is a computer that is far more versatile than most of the "designed down" consumer computers.

It should find many willing takers in industrial control and fluctuate applications. the original microprofessor has a user base of around 3000 and its own user club. The MPF-1 Plus looks as if it should enjoy a similar success amongst those who want to learn how to do things with their micro.

TABLE 1

Category	Command	Function
*Major Function Entry	RESET	Enter and initialize the monitor
	Q	Re-enter the monitor
	E	Enter and initialize the text editor
	R	Re-enter the text editor
	A	Enter two pass assembler
	L	Enter one pass assembler
	D	Enter disassembler
	B	Enter the BASIC language
	C	Re-enter BASIC
Fill in Data	F	Store data in the RAM buffer
Jump Relative	J	Calculate the relative address
Insert Data	I	Insert the contents of a memory block into the RAM
Delete Data	D	Delete one byte of data from the memory
Execution	G	Execute a program which starts from a specified address
Step	S	Single-step a program (Execute a program instruction by instruction).
Display/Alter Registers	R	Display the contents of registers Display the contents of the next pairs of registers Display the contents of the register pairs that precedes the registers currently displayed
	:	Change the contents of registers
Display/Alter Memory	M	Display the contents of specified memory locations Display the contents of the next four bytes Display the contents of the four bytes that precede the current displayed location
	:	Alter the contents of specified memory
	/	Move the contents of a memory block to another location
Manipulate Breakpoint	B	Set or clear breakpoint
Load/Dump Memory	L W	Load data from tape to memory Write data from memory to tape

Our thanks go to Flight Electronics who loaned us the review machine.



EPROM PROGRAMMER FOR THE BBC MICRO

In part two of this project, Peter Simpson and Brian Alderwick describe the software that complements the hardware described last month.

The majority of the 'hard work' involved in programming an EPROM is taken care of by the software. At the heart of the system software are a number of machine code routines and these warrant close examination.

The simplest function is to transfer the contents of the EPROM into the buffer (subroutine <ctb>). Firstly, subroutine <reset> sets the 6821 PIAs to a known state which allows data to be read from the EPROM and sets the address lines into the EPROM to zero. Next, subroutine <barcs> sets the chip select line low which enables data to be read from the EPROM. The accumulator is then loaded with the data byte given out by the EPROM in response to the address on its address lines and this byte is then stored in the buffer. Subroutine <incadd> increments all addresses and checks whether the whole EPROM has been read. If this is not so then the program loops back to loading the accumulator with the data byte given by the new address. If the whole EPROM has been read then the carry flag will be set on return from subroutine <incadd> and subroutine <cs> will be called to set the chip select line high before returning control to the BASIC program.

Verifying the EPROM (subroutine <v>) is very similar to reading, except that an error indication is returned to the BASIC program to show whether a discrepancy was detected between the buffer and the EPROM. The error indication is returned in location "failed". If zero is returned then no discrepancy was found, if non zero then an error was found at the current address. When an error is found the X and Y registers are saved in locations "x" and "y" before returning to the BASIC program which prints an error message and returns control to the machine code at position "vcont". This entry point restores the X and Y registers from locations "x" and "y" and then jumps back into the main loop at address "v2" where the counters are incremented and verifying continued.

Programming Procedures

Programming the EPROM has much in common with reading and verifying notably use of the <reset>, <barcs>, <cs> and <incadd> subroutines. As mentioned earlier however, programming the EPROM requires 50 ms pulses to be applied to the programming pin. The timing for these pulses is derived from a timer in the user 6522 chip which resides in the BBC model B and which is connected to the BBC interrupt system.

The machine operating system (MOS) already has an extensive interrupt routine to handle devices such as the A to D converter, the 6845 CRT controller, the sound chip and the MOS 6522 which controls the system clock. As it stands however, the interrupt system cannot deal with interrupts from the timers in the user 6522 and it is necessary to extend the interrupt routine with a piece of user supplied code which is able to handle these extra interrupts. Acorn have provided an elegant way of adding extra code to the interrupt routine in the form of the IRQ1V and IRQ2V vectors located at &204-5 and &206-7. These vectors hold two byte addresses to which the interrupt routine will jump before and after testing for all IRQs known to the MOS. Since time is not critical in this application the contents of IRQ2V are changed to point at the extra code for dealing with user 6522 timer interrupts. This is achieved by subroutine <inten> which also enables the user 6522 timer to generate interrupts and copies the original contents of IRQ2V into locations IRQRET so that the extra code can return control to the original MOS interrupt routine on completion. Subroutine <intdis> performs the complimentary function of disconnecting the extra code from the interrupt system. The instructions from "int" to "int1" are the extra code added to the interrupt system. The code tests whether the user 6522 timer is the cause of the interrupt. If it is then a value 1 is stored in location "ready" before returning to the MOS routine. If it is not then control is passed directly back to the MOS routine without changing the contents of "ready".

Programming the EPROM, subroutine

PROGRAM 1

```
   10 REM*******************************
   20 REM*                             *
   30 REM*           EPROM             *
   40 REM*         PROGRAMMER          *
   50 REM*          (C) 1983           *
   60 REM*       P.W.G. SIMPSON        *
   70 REM*            and              *
   80 REM*       B.V. ALDERWICK        *
   90 REM*                             *
  100 REM*******************************
  110
  120 MODE7
  130 LOMEM=&3C00
  140 PROCtitle
  150 IF PAGE>&1900 THEN PRINT''
      "PAGE = ";~PAGE'
      "PAGE IS SET TOO HIGH (MAX &1900)"
      ':END
  160 PROCsetup
  170 PROCassemble
  180 CALLreset
  190 PRINT"Loading next program ..."'
  200 Z%=INKEY(200)
  205 PAGE=&1100
  210 HIMEM=&39FF
  220 CHAIN"EPROM2"
  230 END
  240
  250 DEFPROCtitle
  260 DATA EPROM,PROGRAMMER
  270 DATA P.W.G. Simpson PhD
  280 DATA and B.V. Alderwick
  290 FOR I%=1 TO 4
  300 READ T$
  310 PROCwrite
  320 VDU157:PRINT
  330 NEXT I%
  340 ENDPROC
  350
  360 DEFPROCwrite
  370 FOR J%=1 TO 2
  380 A%=16-LEN(T$)/2
  390 VDU157,141,128+I%:PRINTTAB(A%);T$
  400 NEXT J%
  410 ENDPROC
  420
  430 DEFPROCsetup
  440 codeloc%=&3A00
  450 eprom_address_low=&FCF0
  460 IC3_control_1=&FCF1
  470 eprom_address_high=&FCF2
  480 IC3_control_2=&FCF3
  490 eprom_data=&FCE0
  500 IC4_control_1=&FCE1
  510 eprom_control=&FCE2
  520 IC4_control_2=&FCE3
  530 buffadd=&70
  540 buffaddl=&70
  550 buffaddh=&71
  560 length=&74
  570 failed=&75
  580 ready=&76
  590 x=&77
  600 y=&78
  610 IRQRET=&79
  620 via4=&FE64
  630 via5=&FE65
  640 viaacr=&FE6B
  650 viapcr=&FE6C
  660 viaifr=&FE6D
  670 viaier=&FE6E
  680 IRQ2V=&206
  690 oswrch=&FFEE
  700 osbyte=&FFF4
  710 ENDPROC
  720
  730 DEFPROCassemble
  740 PRINT''
      "Assembling machine code ..."
  750 FOR I%=0 TO 2 STEP 2
  760 P%=codeloc%
  770 [         OPT I%
  780 \---------------------------------
  790 \
  800 \ RESET INTERFACE AND COUNTERS
  810 \
  820 .reset     LDA #0
  830            STA IC3_control_1
  840            STA IC3_control_2
  850            STA IC4_control_1
  860            STA IC4_control_2
  870            STA eprom_data
  880            LDA #&FF
  890            STA eprom_address_low
  900            LDA #&3F
  910            STA eprom_address_high
  920            LDA #&83
  930            STA eprom_control
  940            LDA #4
  950            STA IC3_control_1
  960            STA IC3_control_2
  970            STA IC4_control_1
  980            STA IC4_control_2
  990            LDA #3
 1000            STA eprom_control
 1010            LDA #0
 1020            STA eprom_address_low
 1030            STA eprom_address_high
 1040            STA buffaddl
 1050            LDA #&3C
 1060            STA buffaddh
 1070            LDX #0
 1080            LDY #0
 1090            RTS
 1100 \
 1110 \---------------------------------
 1120 \
 1130 \ TRANSFER EPROM TO BUFFER
 1140 \
 1150 .ctb       JSR reset
 1160            JSR barcs
 1170 .ctbl      LDA eprom_data
 1180            STA (buffadd),Y
 1190            JSR incadd
 1200            BCC ctbl
 1210            JSR cs
 1220            RTS
 1230 \
 1240 \---------------------------------
 1250 \
 1260 \ VERIFY EPROM AGAINST BUFFER
 1270 \
 1280 .v         JSR reset
 1290            JSR barcs
 1300 .v1        LDA eprom_data
 1310            SEC
 1320            SBC (buffadd),Y
 1330            BNE v3
 1340 .v2        JSR incadd
 1350            BCC v1
 1360            LDA #0
 1370 .v3        STA failed
 1380            STX x
 1390            STY y
 1400            JSR cs
 1410            RTS
 1420 .vcont     JSR barcs
 1430            LDX x
 1440            LDY y
 1450            JMP v2
 1460 \
 1470 \---------------------------------
 1480 \
 1490 \ PROGRAM EPROM WITH BUFFER
 1500 \
 1510 \ ABANDON ROUTINE IF ESCAPE IS
 1520 \ PRESSED
 1530 \
 1540 .btc       JSR reset
 1550            JSR erased
 1560            LDA failed
 1570            BNE btc4
 1580            JSR setop
 1590            JSR vppon
 1600            JSR inten
 1610            JSR barcs
 1620 .btc1      TXA
 1630            PHA
 1640            LDA #121
 1650            LDX #&F0
 1660            JSR osbyte
 1670            CPX #0
 1680            BPL btc2
 1690            PLA
 1700            JMP btc3
 1710 .btc2      PLA
 1720            TAX
 1730            LDY #0
 1740            LDA (buffadd),Y
 1750            STA eprom_data
 1760            JSR delay
 1770            JSR incadd
 1780            BCC btc1
 1790 .btc3      JSR vppoff
 1800            JSR cs
 1810            JSR intdis
 1820            JSR reset
 1830 .btc4      RTS
 1840 \
 1850 \---------------------------------
 1860 \
 1870 \ CONFIGURE DATA LINES AS
 1880 \ OUTPUTS
 1890 \
 1900 .setop     LDA #0
 1910            STA IC4_control_1
 1920            LDA #&FF
 1930            STA eprom_data
 1940            LDA #4
 1950            STA IC4_control_1
 1960            RTS
 1970 \
 1980 \---------------------------------
 1990 \
 2000 \ GENERATE 50 MILLI-SECONDS
 2010 \ PROGRAMMING PULSE THEN RETURN
 2020 \
 2030 .delay     LDA #0
 2040            STA ready
 2050            JSR barpg
 2060            LDA #80
 2070            STA via4
 2080            LDA #195
 2090            STA via5
 2100 .d1        LDA ready
 2110            CMP #1
 2120            BNE d1
 2130            LDA buffaddl
 2140            CMP #&FF
 2150            BNE d2
 2160            LDA #65
 2170            JSR oswrch
 2180 .d2        JSR pg
 2190            RTS
 2200 \
 2210 \---------------------------------
 2220 \
 2230 \ ENABLE 6522 TIMER INTERRUPT
 2240 \ HANDLER
 2250 \
 2260 .inten     SEI
 2270            LDA IRQ2V
 2280            STA IRQRET
 2290            LDA IRQ2V+1
 2300            STA IRQRET+1
 2310            LDA #(int MOD 256)
 2320            STA IRQ2V
 2330            LDA #(int DIV 256)
 2340            STA IRQ2V+1
 2350            LDA #0
 2360            STA viaacr
 2370            STA viapcr
 2380            LDA #&C0
 2390            STA viaier
 2400            CLI
 2410            RTS
 2420 \
 2430 \---------------------------------
 2440 \
 2450 \ 6522 TIMER INTERRUPT ROUTINE
 2460 \
 2470 .int       LDA viaifr
 2480            BPL int1
 2490            AND #&40
 2500            BEQ int1
 2510            LDA #1
 2520            STA ready
 2530            LDA via4
 2540 .int1      JMP (IRQRET)
 2550 \
 2560 \---------------------------------
 2570 \
 2580 \ DISABLE 6522 TIMER INTERRUPT
 2590 \ HANDLER
 2600 \
 2610 .intdis    SEI
 2620            LDA IRQRET
 2630            STA IRQ2V
 2640            LDA IRQRET+1
 2650            STA IRQ2V+1
 2660            CLI
 2670            RTS
 2680 \
 2690 \---------------------------------
 2700 \
 2710 \ INCREMENT ALL ADDRESS COUNTERS
 2720 \
 2730 .incadd    INC eprom_address_low
 2740            INC buffaddl
 2750            CLC
 2760            BNE ial
 2770            INC eprom_address_high
 2780            INC buffaddh
 2790            INX
 2800            CPX length
 2810            CLC
 2820            BNE ial
 2830            SEC
 2840 .ial       RTS
 2850 \
 2860 \---------------------------------
 2870 \
 2880 \ SET CHIP SELECT LINE LOW
 2890 \
 2900 .barcs     LDA eprom_control
 2910            AND #&81
 2920            STA eprom_control
 2930            RTS
 2940 \
 2950 \---------------------------------
 2960 \
 2970 \ SET CHIP SELECT LINE HIGH
 2980 \
 2990 .cs        LDA eprom_control
 3000            AND #&83
 3010            ORA #2
 3020            STA eprom_control
 3030            RTS
 3040 \
 3050 \---------------------------------
 3060 \
 3070 \ SET CHIP PROGRAMMING LINE LOW
 3080 \
 3090 .barpg     LDA eprom_control
 3100            AND #&82
 3110            STA eprom_control
 3120            RTS
 3130 \
 3140 \---------------------------------
 3150 \
 3160 \ SET CHIP PROGRAMMING LINE HIGH
 3170 \
 3180 .pg        LDA eprom_control
 3190            AND #&83
 3200            ORA #1
 3210            STA eprom_control
 3220            RTS
 3230 \
 3240 \---------------------------------
 3250 \
 3260 \ SWITCH PROGRAMMING VOLTAGE ON
 3270 \ (21 VOLTS)
 3280 \
 3290 .vppon     LDA eprom_control
 3300            AND #&83
 3310            ORA #&80
 3320            STA eprom_control
 3330            RTS
 3340 \
 3350 \---------------------------------
 3360 \
 3370 \ SWITCH PROGRAMMING VOLTAGE OFF
 3380 \
 3390 .vppoff    LDA eprom_control
 3400            AND #&3
 3410            STA eprom_control
 3420            RTS
 3430 \
 3440 \---------------------------------
 3450 \
 3460 \ CHECK THAT EPROM IS FULLY ERASED
 3470 \ NAMELY THAT ALL BYTES ARE &FF
 3480 \
 3490 .erased    JSR barcs
 3500 .eper1     LDA eprom_data
 3510            SEC
 3520            SBC #&FF
 3530            BNE eper2
 3540            JSR incadd
 3550            BCC eper1
 3560            JSR reset
 3570            LDA #0
 3580 .eper2     STA failed
 3590            JSR cs
 3600            RTS
 3610 \
 3620 \---------------------------------
 3630 ]
 3640 NEXT
 3650 PRINT'
      "Machine code located from &";
      ~codeloc%;" to &";~P%-1
 3660 D%=reset
 3670 E%=btc
 3680 F%=ctb
 3690 G%=v
 3700 H%=vcont
 3710 ENDPROC
>
```

<btc>, proceeds as follows. After calling the <reset> subroutine the EPROM is tested to see whether it is fully erased. Contrary to expectation a fully erased EPROM has &FF stored at every location and if this condition is not met then a non zero value is stored at location "failed" which is tested on return from the subroutine. Next subroutine <setop> is called to reset the data lines from inputs to outputs thus enabling data to be written into the EPROM. Next the 21 volt programming voltage is switched on by subroutine <vppon>, the extra interrupt handling routine is connected by subroutine <inten> and the chip select line is taken low by subroutine <barcs>. The main loop starts at this point by getting the contents of the first buffer location and putting it on the EPROM data lines, followed by the generation of a 50ms low pulse on the programming line by subroutine <delay>.

PROGRAM 2

```
10 REM******************************
20 REM*                            *
30 REM*          EPROM             *
40 REM*        PROGRAMMER          *
50 REM*        (C) 1983            *
60 REM*     P.W.G. SIMPSON         *
70 REM*          and               *
80 REM*      B.V. ALDERWICK        *
90 REM*                            *
100 REM******************************
110
120 ON ERROR GOTO 130
130 VDU26
140 CLS
150 PROCsetup
160 PROCvppoff
170 PROCfilesystem
180 PROCselect
190 IF T%=0 THEN PRINTCHR$(129);
    "You must select an EPROM type!"
    :A%=INKEY(200):GOTO180
200 ON ERROR GOTO 4490
210 REPEAT
220 PROCmenu
230 A%=FNoption(10)
240 ON A% GOSUB 280,290,300,310,320,
    330,340,350,360,370
250 UNTIL FALSE
260 END
270
280 PROCselect:RETURN
290 PROCfiletobuff:RETURN
300 PROCbufftofile:RETURN
310 PROCbufftoeprom:RETURN
320 PROCepromtobuff:RETURN
330 PROCverify:RETURN
340 PROCdumpbuff:RETURN
350 PROCsetbuff:RETURN
360 PROCchangebuff:RETURN
370 VDU26:CLS:
    PRINT"EPROM programmer ended":END
380
390 DEFPROCfilesystem
400 REM
410 REM GET FILING SYTEM TYPE.
420 REM SEE USER GUIDE PAGE 454
430 REM
440 X%=&70
450 Y%=0
460 A%=0
470 A%=USR(osargs) AND &FF
480 IF A%=0 OR A%=3 OR A%=6 THEN PRINT
    CHR$(129)"THIS PROGRAM WILL NOT
    WORK WITH YOUR"'CHR$(129)
    "PRESENT FILING SYSTEM":END
490 IF A%=1 OR A%=2 THEN F$="TAPE "
500 IF A%=4 THEN F$="DISK "
510 IF A%=5 THEN F$="ECONET "
520 ENDPROC
530
540 DEFFNoption(A%)
550 REM
560 REM RETURN A VALID OPTION NUMBER
570 REM
580 VDU28,0,22,39,(7+A%),30
590 CLS
600 INPUT"Enter option number ",B%
610 IF B%<=0 OR B%>A% THEN 590
620 =B%
630
640 DEFPROCerase
650 REM
660 REM ERASE PREVIOUS LINE
670 REM
680 VDU11:PRINTSPC(40):VDU11,11
690 ENDPROC
700
710 DEFPROCmenu
720 REM
730 REM PRINT MAIN MENU
740 REM
750 VDU 15,26
760 PRINTTAB(2,24)CHR$(131)
    "ESCAPE will return to main menu";
770 T$="MENU"
780 PROCtitle
790 PRINTTAB(M%)"1)";CHR$(131);
    "Select EPROM type"
800 PRINTTAB(M%)"2) Transfer "+F$+
    "to buffer"
810 PRINTTAB(M%)"3) Transfer buffer
    to "+F$
820 PRINTTAB(M%)"4)";CHR$(129);
    "Transfer buffer to EPROM"
830 PRINTTAB(M%)"5)";CHR$(134);
    "Transfer EPROM to buffer"
840 PRINTTAB(M%)"6)";CHR$(130);
    "Verify EPROM against buffer"
850 PRINTTAB(M%)"7)";CHR$(133);
    "Formatted dump of buffer"
860 PRINTTAB(M%)"8) Load area of
    buffer with char"
870 PRINTTAB(M%)"9) Change one buffer
    location"
880 PRINTTAB(M%-1)"10)";CHR$(129);
    "EXIT"
890 ENDPROC
900
910 DEFPROCsetup
920 REM
930 REM CHECK THAT MACHINE CODE IS
940 REM PRESENT AND INITIALISE
950 REM VARIABLES.
960 REM
970 J%=0
980 FOR I%=&3A00 TO &3BAD
990 J%=J%+?I%
1000 NEXT I%
1010 IF J%=56022 THEN 1030
1020 PRINT''CHR$(129);
     "MACHINE CODE NOT SET UP."''
     CHR$(129);"NOW LEAVING PROGRAM."
     : END
1030 T%=0
1040 reset=D%
1050 btc=E%
1060 ctb=F%
1070 v=G%
1080 vcont=H%
1090 T$=STRING$(40," ")
1100 M%=5
1110 CALL reset
1120 d10=&FCE2
1130 buffadd=&70
1140 !buffadd=0
1150 length=&74
1160 failed=&75
1170 osfile=&FFDD
1180 osargs=&FFDA
1190 REM
1200 REM RESERVE SPACE FOR OSFILE
1210 REM PARAMETER BLOCK
1220 REM
1230 DIM osfilespace 40
1240 ENDPROC
1250
1260 DEFPROCvppoff
1270 REM
1280 REM DONT RETURN UNTIL VPP IS OFF.
1290 REM
1300 A%=?d10:IF (A% AND 4)=0 THEN 1340
1310 PRINT''CHR$(131);
     "Please switch Vpp off."
1320 A%=?d10:IF (A% AND 4)=4 THEN 1320
1330 PRINT'"THANK YOU!":A%=INKEY(100)
1340 ENDPROC
1350
1360 DEFPROCtitle
1370 REM
1380 REM PRINT TITLE IN DOUBLE HEIGHT.
1390 REM
1400 VDU28,0,22,39,0:CLS
1410 Z%=19-LEN(T$)/2
1420 FOR I%=1 TO 2
1430 PRINTSPC(Z%);CHR$(141);T$
1440 NEXT I%
1450 VDU28,0,22,39,6
1460 ENDPROC
1470
1480 DEFPROCselect
1490 T$="SELECT TYPE OF EPROM"
1500 PROCtitle
1510 PRINTTAB(M%)"1) 2764"
1520 PRINTTAB(M%)"2) 27128"
1530 PRINTTAB(M%)"3)"CHR$(129);"EXIT"
1540 A%=FNoption(3)
1550 T%=0
1560 ON A% GOTO  1570,1580,1610
1570 T%=&2000:PRINT'
     "Type chosen = 2764.":GOTO1590
1580 T%=&4000:PRINT'
     "Type chosen = 27128."
1590 IF NOT FNok THEN 1540
1600 ?length=T% DIV 256
1610 ENDPROC
1620
1630 DEF FNok
1640 REM
1650 REM RETURN TRUE/FALSE IF
1660 REM CORRECT/NOT CORRECT.
1670 REM
1680 LOCAL T$
1690 PRINT'''
1700 REPEAT
1710 *FX15,1
1720 PROCerase
1730 INPUT"Is this correct (Y/N) ",T$
1740 UNTIL T$="Y" OR T$="N"
1750 IF T$="Y" THEN =TRUE ELSE =FALSE
1760
1770 DEFPROCfiletobuff
1780 T$="TRANSFER "+F$+"TO BUFFER"
1790 PROCtitle
1800 PRINTTAB(M%)"1) Transfer data"
1810 PRINTTAB(M%)"2)";CHR$(129);"EXIT"
1820 A%=FNoption(2)
1830 ON A% GOTO 1840,2150
1840 PRINT''
1850 PRINT"Name of "+F$+"file";
     :INPUT N$
1860 INPUT"Buffer load address (HEX)
     &"T$
1870 L%=EVAL("&"+T$)
1880 CLS
1890 PRINT"File name = ";N$''
     "Load address = &";~L%
1900 IF NOT FNok THEN 1820
1910 IF NOT FNaddok(L%,L%) THEN 1820
1920 REM
1930 REM GET FILE CATALOGUE INFORMATION
1940 REM INCLUDING FILE LENGTH.
1950 REM THIS DOES NOT WORK WITH THE
1960 REM TAPE FILING SYSTEM SO IGNORE
1970 REM
1980 IF F$="TAPE " THEN 2080
1990 PROCosfile(5,N$,0,0,0,0)
2000 IF NOT ok THEN 1820
2010 FL%=osfilespace!10
2020 EA%=L%+FL%-1
2030 REM
2040 REM CHECK THAT FILE WILL FIT
2050 REM INSIDE BUFFER.
2060 REM
2070 IF NOT FNaddok(L%,EA%) THEN 1820
2080 L%=L%+&3C00
2090 IF NOT FNprocede THEN 1820
2100 REM
2110 REM LOAD FILE
2120 REM
2130 PROCosfile(&FF,N$,L%,0,0,0)
2140 PRINT'CHR$(130);
     "FINISHED":PROCwait
2150 ENDPROC
2160
2170 DEFFNaddok(X%,Y%)
2180 REM
2190 REM CHECK FOR VALID BUFFER ADDRESS
2200 REM
2210 IF Y%>&3FFF THEN PRINT'CHR$(129);
     "Finish address outside buffer."
     :GOTO2240
2220 IF Y%<X% THEN PRINT'CHR$(129);
     "End address less than start
     address.":GOTO2240
2230 =TRUE
2240 PROCwait
2250 =FALSE
2260
2270 DEFPROCwait
2280 REM
2290 REM WAIT UNTIL SPACE BAR IS
2300 REM PRESSED
2310 REM
2320 VDU15
2330 PRINT'"Press space bar to return
     to menu."
2340 *FX15,1
2350 IF GET$<>" " THEN 2350
2360 ENDPROC
2370
2380 DEFPROCosfile(A%,T$,LO%,EX%,ST%,
     EN%)
2390 REM
2400 REM SET UP PARAMETER BLOCK FOR
2410 REM OSFILE MOS CALL.
2420 REM
2430 IF LEN(T$)<21 THEN 2480
2440 PRINTCHR$(129);
     "File name too long."
2450 PROCwait
2460 ok=FALSE
2470 ENDPROC
2480 C%=0
2490 X%=osfilespace MOD 256
2500 Y%=osfilespace DIV 256
2510 P%=osfilespace+18
2520 $P%=T$
2530 osfilespace!0=P%
2540 osfilespace!2=LO%
2550 osfilespace!6=EX%
2560 osfilespace!10=ST%
2570 osfilespace!14=EN%
2580 A%=USR(osfile) AND &FF
2590 ok=TRUE
2600 IFA%<>0 OR F$="TAPE " THEN2640
2610 PRINT'CHR$(129)"FILE NOT FOUND"
2620 ok=FALSE
2630 PROCwait
2640 ENDPROC
2650
2660 DEFFNprocede
2670 REM
2680 REM RETURN TRUE/FALSE ACCORDING
2690 REM TO PROCEDE/NOT PROCEDE.
2700 REM
2710 LOCAL T$
2720 PRINT
2730 REPEAT
2740 *FX15,1
2750 PROCerase
2760 INPUT"Do you wish to procede
     (Y/N) ",T$
2770 UNTIL T$="Y" OR T$="N"
2780 IF T$="Y" THEN =TRUE ELSE =FALSE
2790
2800 DEFPROCbufftofile
2810 T$="TRANSFER BUFFER TO "+F$
2820 PROCtitle
2830 PRINTTAB(M%)"1) Transfer data."
2840 PRINTTAB(M%)"2)";CHR$(129);"EXIT"
2850 A%=FNoption(2)
2860 ON A% GOTO 2870,3030
2870 PRINT'
2880 PRINT"Name of "+F$+"file ";
     :INPUT N$
2890 INPUT"Start address &"T$
2900 SA%=EVAL("&"+T$)
2910 INPUT"End address &"T$
2920 EA%=EVAL("&"+T$)
2930 CLS
2940 PRINT"File name = ";N$
2950 PRINT'"Start address = &";
     ~SA%'"Finish address = &";~EA%
2960 IF NOT FNok THEN2850
2970 IF NOT FNaddok(SA%,EA%) THEN 2850
2980 SA%=SA%+&3C00:EA%=EA%+&3C01
2990 IF NOT FNprocede THEN 2850
3000 PROCosfile(0,N$,SA%,SA%,SA%,EA%)
3010 IF NOT ok THEN 2850
3020 PRINT'CHR$(130);"FINISHED"
     :PROCwait
3030 ENDPROC
3040
3050 DEFPROCsetbuff
3060 T$="SET AREA OF BUFFER TO ONE
     CHARACTER":PROCtitle
3070 PRINTTAB(M%)
     "1) Define as an ASCII character"
3080 PRINTTAB(M%)
     "2) Define as a number"
3090 PRINTTAB(M%)"3)";CHR$(129);"EXIT"
3100 A%=FNoption(3)
3110 ON A% GOTO3120,3160,3310
3120 INPUT'"Enter character ",T$
3130 T$=LEFT$(T$,1)
3140 C%=ASC(T$)
3150 GOTO3180
3160 INPUT'"Enter number (HEX) &"T$
3170 C%=EVAL("&"+T$)
3180 INPUT"Enter start address (HEX) &"
     ,T$
3190 SA%=EVAL("&"+T$)
3200 INPUT"Enter end address (HEX) &"
     ,T$
3210 EA%=EVAL("&"+T$)
3220 CLS
3230 PRINT'"Number is HEX &";~C%,
     "Decimal ";C%
3240 IF C%>31 AND C%<127 THEN PRINT
     "Character is "" ";CHR$(C%);" """
3250 PRINT'"Start address = &"
     ;~SA%'"End address = &";~EA%
3260 IF NOT FNok THEN 3100
3270 IF NOT FNaddok(SA%,EA%) THEN 3080
3280 SA%=SA%+&3C00:EA%=EA%+&3C00
3290 IF NOT FNprocede THEN 3100
3300 FOR I%=SA% TO EA%:?I%=C%:NEXT I%
3310 ENDPROC
3320
3330 DEFPROCdumpbuff
3340 T$="FORMATTED DUMP OF BUFFER"
     :PROCtitle
3350 PRINTTAB(M%)"1) Formatted dump"
3360 PRINTTAB(M%)"2)";CHR$(129);"EXIT"
3370 A%=FNoption(2)
3380 ON A% GOTO 3390,3620
3390 INPUT'"Start address (HEX) &",T$
3400 SA%=EVAL("&"+T$)
3410 INPUT"Finish address (HEX) &",T$
3420 EA%=EVAL("&"+T$)
3430 CLS
3440 PRINT'"Start address = &";~SA%'
     "Finish address = &";~EA%
3450 IF NOT FNok THEN 3370
3460 IF NOT FNaddok(SA%,EA%) THEN 3370
3470 VDU28,0,22,39,6,14
3480 CLS
3490 FOR I%=SA% TO EA% STEP 8
3500 @%=4
3510 PRINT~I%;TAB(6);
3520 @%=3
3530 T$=""
3540 FOR J%=&3C00 TO &3C07
3550 A%=J%?I%
3560 PRINT~A%;
3570 IF A%<32 OR A%>126 THEN T$=T$+"."
     ELSE T$=T$+CHR$(A%)
3580 NEXT J%
3590 PRINT" ";T$
3600 NEXT I%
3610 PROCwait
3620 ENDPROC
3630
3640 DEFPROCchangebuff
3650 T$="CHANGE ONE CHARACTER IN
     BUFFER"
3660 PROCtitle
3670 PRINTTAB(M%)
     "1) Define as an ASCII character.
3680 PRINTTAB(M%)
     "2) Define as a number."
3690 PRINTTAB(M%)"3)";CHR$(129);"EXIT"
3700 A%=FNoption(3)
3710 ON A% GOTO 3720,3750,3870
3720 INPUT'"Enter character ",T$
3730 T$=LEFT$(T$,1)
3740 C%=ASC(T$):GOTO3770
3750 INPUT'"Enter number (HEX) &",T$
3760 C%=EVAL("&"+T$)
3770 INPUT"Enter address (HEX) &"T$
3780 I%=EVAL("&"+T$)
3790 CLS
3800 PRINT'"Number is HEX &";~C%,
     "Decimal ";C%
```

```
3810 IF C%>31 AND C%<127 THEN PRINT
     "Character is "" ";CHR$(C%);" """
3820 PRINT'"Address = &";~I%
3830 IF NOT FNok THEN 3700
3840 IF NOT FNaddok(I%,I%) THEN 3700
3850 IF NOT FNprocede THEN 3700
3860 I%?&3C00=C%
3870 ENDPROC
3880
3890 DEFPROCepromtobuff
3900 T$="TRANSFER EPROM TO BUFFER"
3910 PROCtitle
3920 PRINTTAB(M%)"1) Transfer EPROM"
3930 PRINTTAB(M%)"2)";CHR$(129);"EXIT"
3940 A%=FNoption(2)
3950 ON A% GOTO 3960,3990
3960 IF NOT FNprocede THEN  3940
3970 CALL ctb
3980 PRINT'CHR$(130);"FINISHED"
     :PROCwait

3990 ENDPROC
4000
4010 DEFPROCbufftoeprom
4020 T$="TRANSFER BUFFER TO EPROM"
4030 PROCtitle
4040 PRINTTAB(M%)"1) Transfer buffer"
4050 PRINTTAB(M%)"2)";CHR$(129);"EXIT"
4060 A%=FNoption(2)
4070 ON A% GOTO 4080,4220
4080 IF NOT FNprocede THEN 4060
4090 CALL reset
4100 PRINT'CHR$(131);
     "Please switch Vpp on."
4110 REPEAT
4120 A%=?d10
4130 UNTIL (A% AND 4)=4
4140 PRINT'"THANK YOU! Running....."

4150 X%=VPOS+1:PRINT'
     STRING$((T% DIV 256),".");
     TAB(0,X%);
4160 ?length=T% DIV 256
4170 CALL btc

4180 IF ?failed=0 THEN 4200
4190 PRINT''CHR$(129);"EPROM not
     erased at address &";
     ~!buffadd-&3C00
4200 SOUND1,-15,100,30:PROCvppoff
4210 PRINT'CHR$(130);"FINISHED"
     :PROCwait
4220 ENDPROC
4230
4240 DEFPROCverify
4250 ok=TRUE
4260 T$="VERIFY EPROM AGAINST BUFFER"
4270 PROCtitle
4280 PRINTTAB(M%)"1) Verify EPROM"
4290 PRINTTAB(M%)"2)";CHR$(129);"EXIT"

4300 A%=FNoption(2)
4310 ON A% GOTO4320,4460
4320 VDU14
4330 IF NOT FNprocede THEN 4300
4340 CALL v
4350 IF ?failed=0 THEN 4430
4360 PRINT
4370 ok=FALSE
4380 REPEAT
4390 PRINT"Failed at address &";
     ~!buffadd-&3C00
4400 CALL vcont
4410 UNTIL ?failed=0
4420 VDU15
4430 PRINT'CHR$(130);"VERIFY";
4440 IF ok THEN PRINT" OK"
     :ELSE PRINT CHR$(129);"FAILED"
4450 PRINT'CHR$(130);"FINISHED"
     :PROCwait
4460 ENDPROC
4470
4480 REM ERROR HANDLING ROUTINE
4490 VDU15
4500 IF ERR=28 THEN PRINT'CHR$(129);
     "What you just entered was not a"
     CHR$(129);"hex number!":PROCwait
     :GOTO210
4510 IF ERR=17 THEN PRINT''CHR$(131);
     "You pushed ESCAPE":PROCvppoff
     :PROCwait:GOTO210
4520 REPORT:PRINT:PROCwait:GOTO210
>
```

The 50ms pulse is generated by setting the programming line low using subroutine <barpg>, then instructing the 6522 timer to generate an interrupt after 50ms. Having given this instruction the computer then goes into a loop which examines the value of location "ready". The computer will remain in the loop until location "ready" has the value 1 and the only way it can obtain this value is when the extra interrupt code is executed following an interrupt by the 6522. Thus the computer will remain locked in the loop until the 6522 timer generates an interrupt ie for 50ms. After this the programming line is set high and control returned to subroutine <btc>. Addresses are incremented and the loop continues in the normal way until all locations in the EPROM have been programmed. The chip select line is then taken high by subroutine <cs>, the programming voltage is switched off by subroutine <vppoff> and the extra interrupt code is disconnected by subroutine <intdis>.

Down The Tube?

As written the program will not run on the second processor via the tube. This decision was not taken lightly and was principally influenced by the facts that the resultant machine code program was both smaller and faster than its tube compatible equivalent. The entire source code for the machine code program has been deliberately given in figure 7 so that those wishing to develop a tube compatible version will be able to do so with the minimum of difficulty.

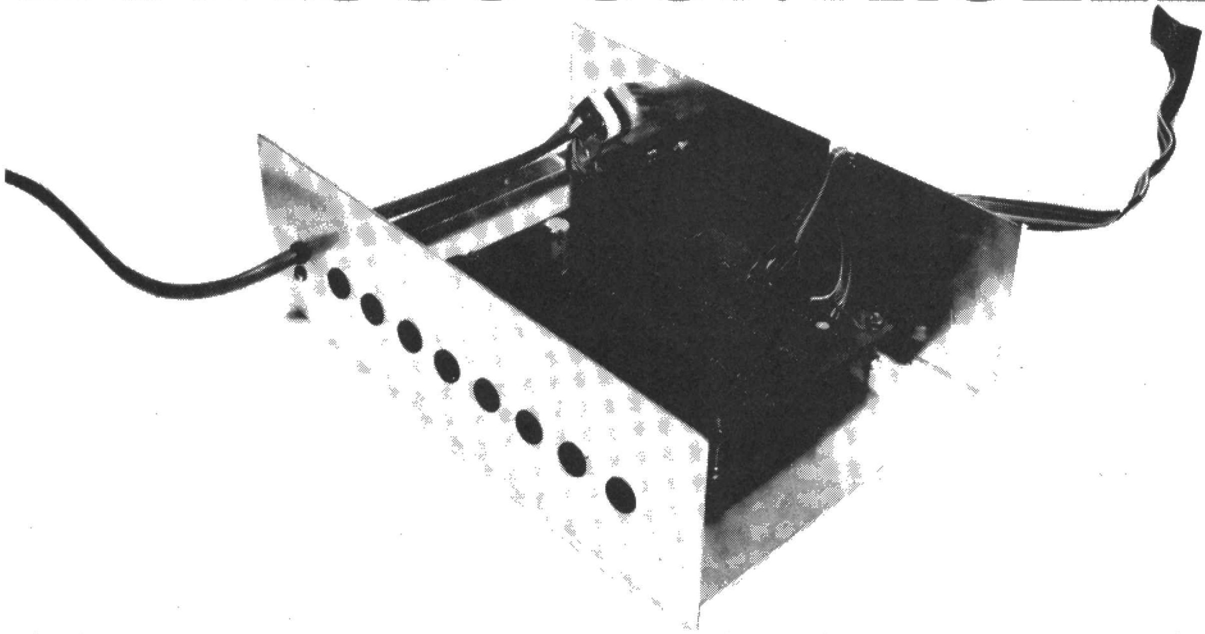
In conclusion, the programmer has been designed to provide a cost effective way of programming your own EPROMs and the software has been written to minimise user errors. Happy programming!

Copies of the software are available from the authors at 40 The Chase, Cashe's Green, Stroud, Gloucs, GL5 4SB.

The cost is £7.00 inclusive. Please state whether a cassette, 40 track disk or 80 track disk is required.

E&CM

ZX DISCO CONTROLLER



Mark Stuart presents a project that enables ZX computers to drive eight individual banks of mains lamps.

There have been many designs for effects light controllers published in recent years. Most of these though have been circuits that either flashed lights in time with an audio, usually musical, signal or have offered a pseudo random flash sequence. Introducing a computer into a light controller offers a far wider range of possibilities and the ZX81 and Spectrum computers are ideally suited to the task,

The project described here, in addition to providing on/off control of the lamps, provides a feedback pulse which can be read by the computer each time the AC mains voltage passes through zero.

Straightforward programs written in BASIC can be used for independent on-off control of the lamps. This gives a wide range of very effective strobe, chase, and similar effects while more complicated programs, which use machine code to read the mains 'zero cross' feedback pulse, can be used to vary the brightness of each lamp from zero to full power. By using appropriate colour combinations and imaginative arrangements of the 8 banks of lamps it is possible to produce an enormous variety of light show effects.

Functions And Specification

The computer interface is memory mapped for use with the ZX81. Any address in the range 8K-16K can be POKEd to drive the banks of lamps or PEEKed to read the mains zero crossing pulses. When used with the ZX Spectrum the interface is "I/O mapped". The instructions, OUT 65503 & IN 65503, replace the POKE & PEEK instructions of the ZX81 version.

The 8 data lines drive one bank of lamps each in the usual arrangement of an 8 bit output port. The ZX81 instruction POKE 8192,0 turns all lamps off & POKE 8192,255 turns all lamps on. The techniques for individual control of lamps will be explained later.

When the ZX81 instruction PEEK 8192 is issued the computer will read the state of the AC mains "zero cross" circuit. Data line D0 reads as 1 whenever the mains voltage is below approximately 30 volts, otherwise it reads as 0.

The brightness of the lamps can be controlled individually by referring to the "zero cross" pulse & switching on the appropriate outputs at set points during each mains half cycle. This method of "phase control" provides the same power control as a standard domestic lamp dimmer switch. Each output has a mains voltage triac rated at 5.5A rms. The triacs are not fitted with heatsinks and so should be derated to 1 amp for continuous use.

Isolation between the computer and the mains is ensured by the use of opto-isolators. When the unit is being used with on/off instructions it is desirable to minimise interference by switching power on to the load when the AC mains voltage is at or near to zero. This can be achieved by selecting "Auto zero cross" switching using SW1. Auto zero cross switching delays the switching on of the output triacs until the mains falls to near zero, regardless of when the instruction was received.

Circuit Description

The circuit diagram is shown in **Fig 1.** The triac drive components are shown for only one channel (components with "a" suffix), the other 7 channels are identical.

Address decoding is provided by IC1, IC2, R1 and D1. For the ZX81 computer the lines used are A13, A14, and A15. When A13 is high and A14, A15 and MREQ are all low, IC1c output will go low. This can only occur for addresses between 8192 and 16383. The output of IC1c is also gated with the WR and RD lines by IC1a and IC1b respectively. A valid memory read instruction therefore sends IC1b output low; a write instruction sends IC1a output low. IC2c and IC2d are connected in parallel and pull the ROMCS line high via D1. This ensures that the ROM "echo" between addresses 8192 and 16383 is disabled. For the ZX Spectrum IC2 and D1 are not fitted. IC1d output is disconnected by omitting the link between pins 8 and 12. The output of IC1c goes low when A5 and IORQ are both low. This simple decoding means that the Spectrum will respond to any OUT or IN address that sets A5 low. The highest value, 65503, is used as this leaves all other address lines high. To discriminate between IN and OUT instructions the RD and WR lines respectively are gated with IC1c output in the same way as the ZX81 version.

Memory write instructions (OUT or POKE) to the correct memory or I/O addresses enable IC3, which is an 8 bit parallel latch. The 8 inputs to IC3 are connected to the computer data bus lines D0 to D7. When the latch is enabled the data present on D0 to D7 is latched onto the 8 latch output lines. It will be held there until the next valid POKE or OUT instruction updates it. Each of the latch output lines is connected via a current limiting resistor to the input of an opto isolator. The opto isolator consists of a light emitting diode and a phototransistor encapsulated so that they are electrically insulated from each other. The insulation is transparent so that light from the light emitting diode can pass freely to the phototransistor. When the LED is supplied with current it emits light; this light is received by the phototransistor which becomes conductive. If the current is switched off the LED does not emit light, therefore the phototransistor becomes non conductive.

When the output of IC3 corresponding to D0 is low (logic 0) the LED in opto isolator OP2a does not receive any current and hence the phototransistor is non conductive. Transistor Q3a receives its base drive current via the phototransistor in OP2a and therefore will be turned off. The gate of triac 1a is in turn driven from the collector of Q3a via R8a. Since Q3a is turned off there will be no gate drive to triac 1a which will remain non conductive. If the D0 output of IC3 becomes high (logic 1) then the LED in OP2a is turned on, via current limiting resistor R3a, and the phototransistor conducts. If it is assumed that Q4 is turned on, current passes via Q4 and the opto isolator to the base of Q3a, which is turned on. Current from Q3a passes via current limiting resistor R8a to the gate of triac 1a, which turns on and switched mains voltage ot the lamp.

The phototransistor base is connected to the negative supply via R5a to increase its switching speed. To prevent spurious triggering, resistor R4a reduces the sensitivity of Q3a, and R9a reduces the gate sensitivity of the triac.

Transistors Q4 and Q5 with their associated components provide mains voltage zero crossing detection and switching. The rectified unsmoothed secondary voltage from T1 is applied to the base of Q5 via potential divider resistors R11 and R12. In order to turn on Q5 the voltage at its base must exceed 0.6V. As R11 and R12 are equal the voltage at the top of R11 will be twice that at Q5 base. Whenever the voltage at the top of R11 exceeds 1.2V Q5 is turned on. **Fig 2** shows the associated voltage waveforms.

The collector of Q5 produces positive pulses each time the mains voltage falls below approximately 30 volts. Provided

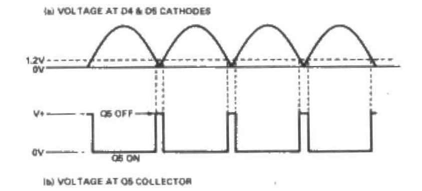


Figure 2. Waveforms associated with the zero crossing detection formed by Q4, Q5 and associated components.

SW1 is open, pulses from Q5 collector pass via D2 to R6 and R7 to switch on Q4 during the zero crossing period. When the phototransistor in OP2a becomes conductive it can only pass current to Q3a whilst Q4 is turned on. In this way it is ensured that no matter when the computer sends a logic 1 to the latch the triac will only receive trigger pulses during the zero crossing period.

This method of switching minimises interference, and improves lamp life because of the much reduced surge current. **Fig 3** shows the improvement graphically. In **Fig 3b** the triac is triggered at the instant the computer gives the switch on instruction. This could be anywhere during the mains cycle but has been shown at the cycle peak which represents the worst case. The voltage across the load instantly rises to the mains peak voltage (350V). This produces a massive surge current, especially with tungsten filament lamps which have a cold resistance of less than one tenth of their operating value. In contrast, the wave form of **Fig 3c** shows the mains voltage rising steadily from zero at the beginning of the mains half cycle following the switch on instruction.

Switch SW1 can be closed to bypass the zero crossing pulses so that the triacs can if required be triggered at random. This mode of operation is essential for computer controlled dimming which will be explained later. Memory read instructions (IN or PEEK) allow the computers to gain access to the state of the zero crossing pulse circuit. A valid read instruction produces a logic 0 on the output of IC1b, which turns off transistor Q1. If Q5 is turned on the LED in the optoisolator is energised. This causes the phototransistor in OP1 to become conductive, providing base current for Q2 which turns on, pulling data line D0 low. The computer will therefore read D0 as a logic 0. The other 7 data lines are left floating, and assume the logic 1 state. During the mains zero crossing period Q5 is turned off and the LED in OP1 does not receive any current. The phototransistor becomes non conductive, Q2 remains off, and so data line D0 is effectively left floating and therefore read by the computer as a logic 1. When the computer is not giving a valid read instruction IC1b output is high and turns on Q1. When Q1 is turned on it short circuits the base-emitter junction of Q2 which therefore is turned off. This ensures that the data line D0 is not loaded.

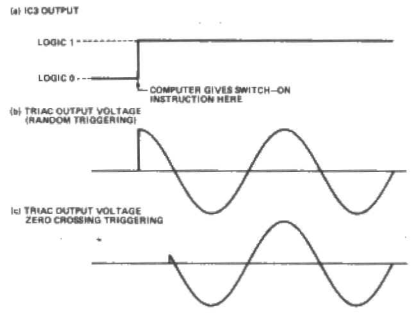


Figure 3. A graphical illustration of the way in which the zero crossing detector can reduce surge currents.

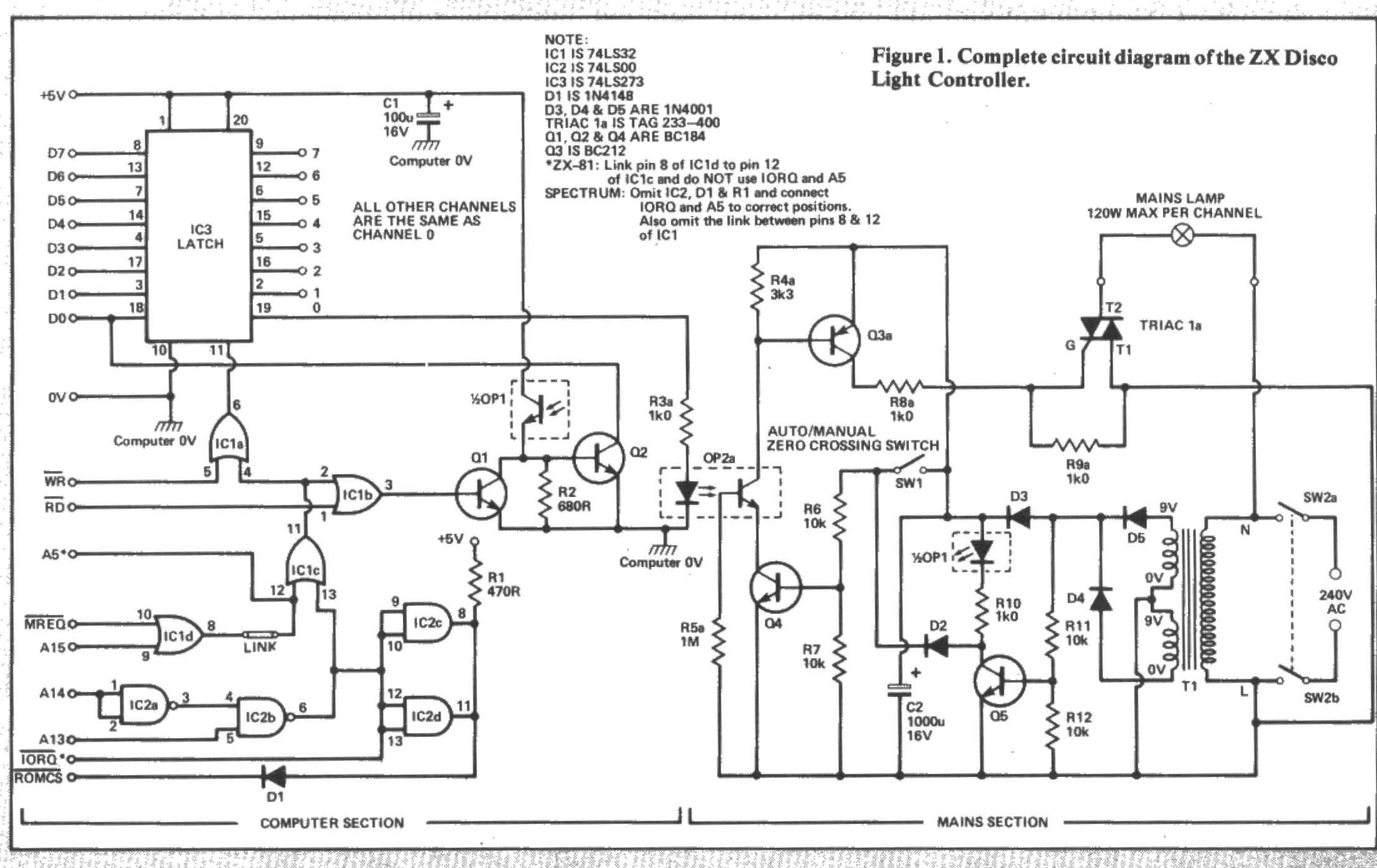


Figure 1. Complete circuit diagram of the ZX Disco Light Controller.

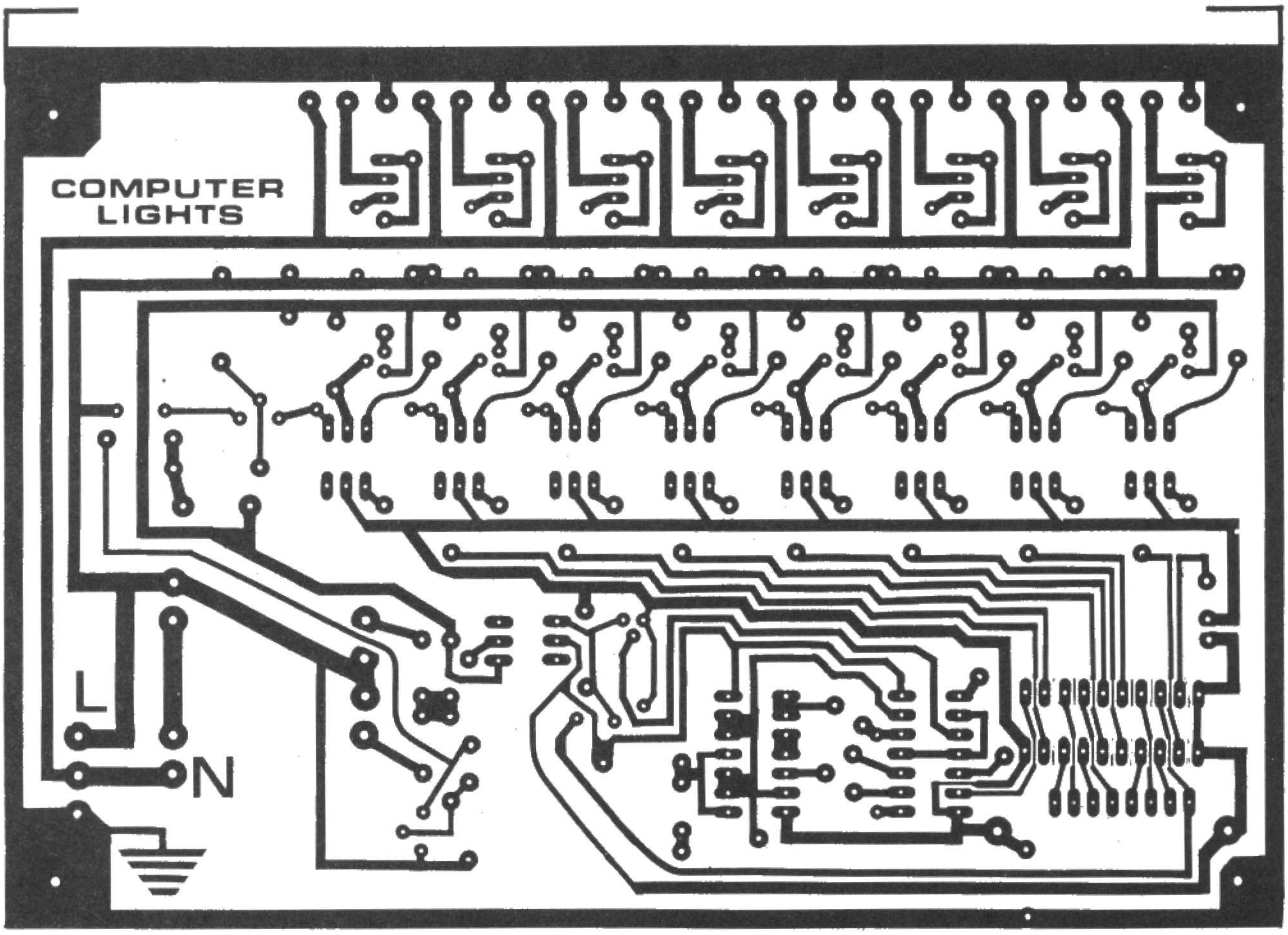


Figure 4. The full size foil pattern of the ZX Light Controllers PCB.

Assembly

The entire unit is built on a single printed circuit board. Assembly of the PCB is straightforward but care should be taken to keep excess solder or component leads from reducing the space between the mains section and the computer section of the circuit. The ICs and opto isolators should be fitted in sockets. If 6 pin sockets cannot be obtained use 8 pin sockets with the 2 end contacts removed – there is enough space for this. On the PCB overlay drawing **Fig 5** only the components for the D0 output channel have been shown. The other 7 channels are identical except for the orientation of R3, which differs slightly.

Take particular care with the triacs which should be mounted with their metal face nearest to R9. SW1 is connected to the PCB via insulated flying leads. Ensure that C1, C2 and the transistors and diodes are fitted the right way round. The specified transformer will only fit one way to the PCB. When the PCB has been assembled, before inserting the opto isolators and ICs, the computer connecting lead should be fitted.

Figure 6 shows the connections between the rear of a ZX81 or Spectrum edge connector and the PCB. Two pieces of 10 way ribbon cable, approximately 50cms long, are used – designated ribbon 1 and 2. It is best to connect the leads to the edge connectors first. Remember that the polarising slot is numbered when counting the pins. It is unlikely that any memory expansion will be required since even simple programs produce excellent effects, and screen

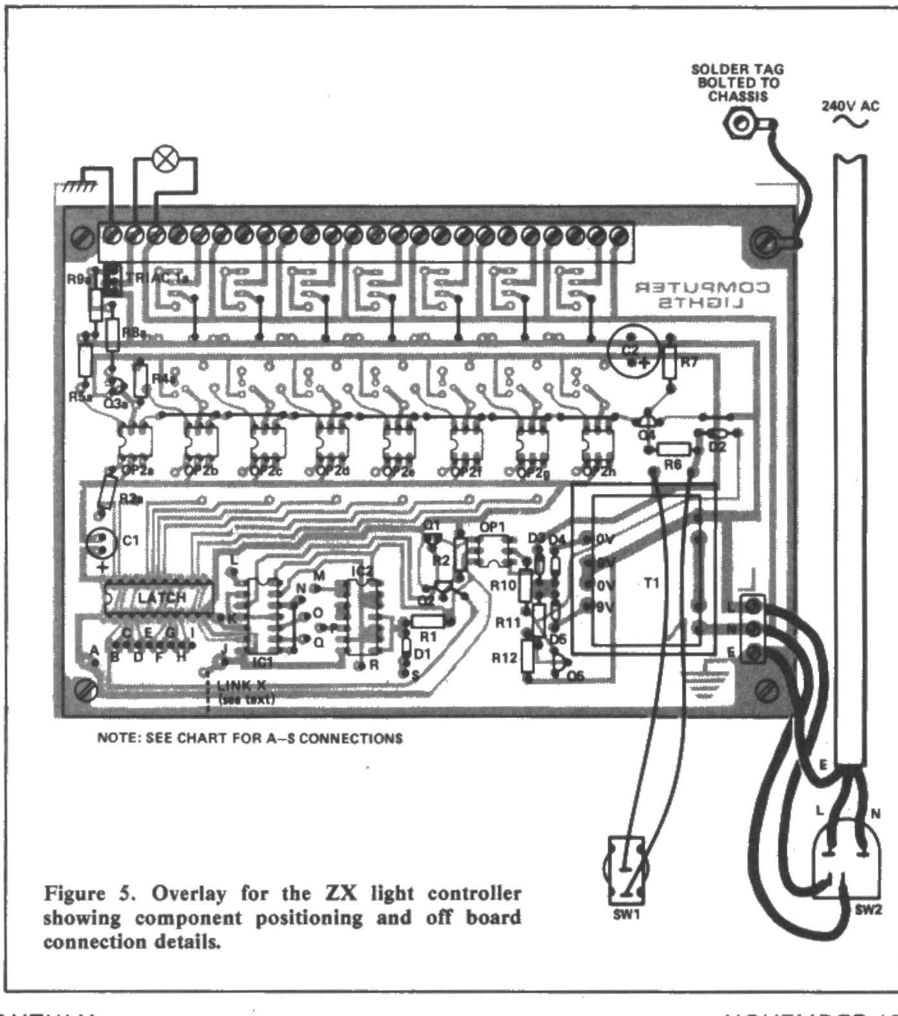
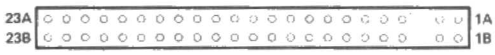


Figure 5. Overlay for the ZX light controller showing component positioning and off board connection details.

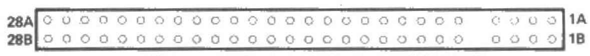
	COLOUR	ZX-81	SPECTRUM	FUNCTION	PCB LETTER
RIBBON 1	BLACK	4A	6A	D0	B
	BROWN	5A	7A	D1	C
	RED	6A	8A	D2	D
	ORANGE	9A	11A	D3	E
	YELLOW	10A	12A	D4	F
	GREEN	8A	10A	D5	G
	BLUE	7A	9A	D6	H
	PURPLE	1A	3A	D7	I
	GREY	1B	3B	+5V	A
	WHITE	4+5B	6+7B	0V	J
RIBBON 2	BLACK	17A	19A	$\overline{WR}$	K
	BROWN	16A	18A	$\overline{RD}$	L
	RED	NC	23B	A5	N*
	ORANGE	14A	16A	$\overline{MREQ}$	O
	YELLOW	11B	1A	A15	Q
	GREEN	12B	1B	A14	M
	BLUE	13B	2A	A13	P
	PURPLE	NC	17A	$\overline{IORQ}$	R*
	GREY	23B	NC	$\overline{ROMCS}$	S

(CONNECTOR: ZX-81 / SPECTRUM)

NC = NOT CONNECTED
* = SPECTRUM ONLY



REAR VIEW OF ZX-81 EDGE CONNECTOR



REAR VIEW OF ZX–SPECTRUM EDGE CONNECTOR

RIBBON CABLE No. 1 HAS 10 LEADS
RIBBON CABLE No. 2 HAS 9 LEADS

Figure 6. Details of the links between the controller and either a ZX81 or Spectrum computer.

memory is not used. For the ambitious however, it is possible to add an edge plug to the rear of the edge connector without difficulty if the wiring is connected close to the base of the pins.

Link X on the PCB may be used to connect the computer power supply negative lead to mains earth. This connection is optional since adequate isolation is provided by the opto isolators. Its use gives an added measure of safety. One disadvantage of using the earth connection is that the saving and loading of programs may be impaired due to hum loops.

Once all soldering to the board is completed it is essential to clean away any flux deposits from beneath the opto isolators. If not removed these deposits cause leakage currents and erratic operation of the outputs can follow. When thoroughly clean and dry the area should be coated with a good insulating lacquer.

The completed PCB is mounted in an earthed metal case. Four sets of long screws should be used and the board should be spaced from the case by approximately 1cm by using extra nuts. This method ensures a good connection between the case and the earth track around the PCB. An additional earth link should be made between a solder tag screwed to the case and the PCB as shown.

A row of 24 terminals on the rear of the PCB provides for the 8 outputs. Holes should be drilled in the rear of the case and fitted with grommets or cable clamps for the 8 outputs and the mains input lead.

The only front panel components are the mains on/off switch SW2 and mode switch SW1.

The incoming earth wire and the mains from SW1 are connected to the remaining three PCB terminals as shown in **Fig 5.** A slot for the ribbon cable should be cut in the top edge of the front panel. Take care to route the cable away from any of the mains carrying parts of the circuit.

Fuses

No provision for fusing has been made on the PCB. A 1 amp fuse must be fitted to each output lead either by fitting a fuseholder inside the case or within the bank of lamps. The incoming mains lead must be fused at 5 amps or less. The overall rating of the unit should be limited to 120 watts per channel. This limit puts very little strain on the triacs and minimises heat build up inside the case.

For those requiring higher output power, it is possible to use the outputs to drive higher power "slave" triacs capable of handling several kilowatts of load.

Testing

The unit should first be checked for safety. Assuming the ICs and opto isolators are not fitted check for insulation between the computer leads and the incoming mains lead using an ohmmeter. Disconnect the mains and check for continuity between the mains earth lead and the case, and the 8 output earth terminals. Connect the mains input and check that the computer lead is completely isolated using an AC voltmeter. Check that all 8 outputs are off by measuring between each pair of L and N output terminals. Measure the voltage between the emitter of Q5 and the cathode of D3, it should be approximately 14 volts.

Assuming all the checks give correct results, switch off, insert the 3 ICs (2 for the Spectrum version), and the opto isolators. Connect the computer.

Switch on the computer and check that it runs normally. Connect a lamp to each output channel and switch on. Some or all of the lamps may light depending upon the state assumed by IC3 when the computer was turned on. To turn all of the lamps off enter the instruction POKE 8192, 0 for the ZX81 or OUT 65503, 0 for the Spectrum. To test each lamp substitute the numbers 1,2,4,8,16, 32, 64, 128 in turn for the 0 in the above instructions. Check that SW1 has no effect on the operation so far.

To check the zero crossing pulse detection circuit run the following simple program:-

```
10 FOR N = 1 TO 100
20 PRINT PEEK 8192;
30 PRINT " ";
40 NEXT N
```

This is the ZX81 version. For the Spectrum line 20 should read:-

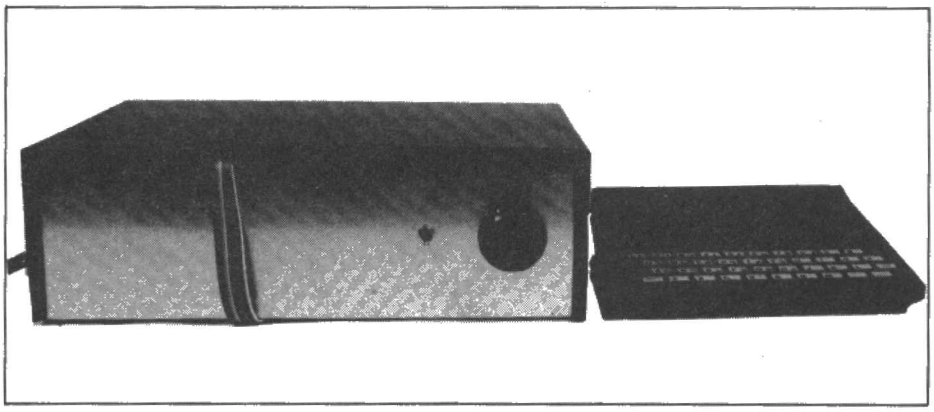
```
PRINT IN 65503;
```

The screen should display 100 numbers. Most of them will be 254 with the occasional 255. Switch off SW1 and run the program again. This time the result should be an unvarying 255. The computer reads 255 when the switch is off because all the data lines float to logic 1. When the switch is on D0 is pulled low the majority of the time, giving decimal 254. D0 is only allowed to float high during the short zero cross period, giving decimal 255. The program reads the data at random and therefore receives a scatter of the two numbers in proportion to the ratio of the pulse width and the mains cycle.

Testing is now complete.

CONTINUED NEXT MONTH

E&CM



THE TOP TEN REVIEWED

This month E&CM compares the performance of ten of the top selling home computers on the market. In particular, our reviewers S. M. Gee, Mike James, and Harry Fairhead examine the potential of these machines for advanced programming and system expansion.

ACORN ELECTRON
Price: £199
Standard memory: 32K
Expandable to: non-expandable

ATARI 400/800
Price: £100/£200
Standard memory: 16K/48K
Expandable to: non-expandable

BBC MICRO MODEL B
Price: £399
Standard memory: 32K
Expandable to: non-expandable

COMMODORE 64
Price: £299
Standard memory: 64K
Expandable to: non-expandable

COMMODORE VIC 20
Price: £150
Standard memory: 5K
Expandable to: 32K

DRAGON 32
Price: £175
Standard memory: 32K
Expandable to: 64K

ORIC
Price: £130/£169
Standard memory: 16K/48K
Expandable to: non-expandable

SINCLAIR SPECTRUM
Price £99/£125
Standard memory: 16K/48K
Expandable to: 48K

SINCLAIR ZX81
Price: £45
Standard memory: 1K + 16K
Expandable to: 56K

TI99/4A
Price: £150
Standard memory: 16K
Expandable to: 52K

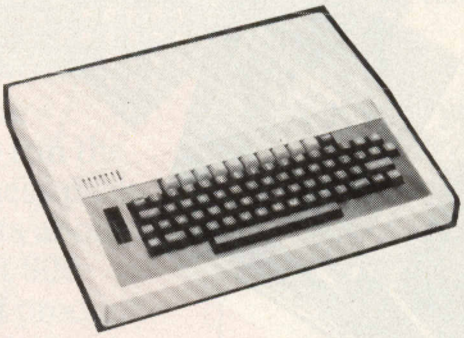
The BBC Micro

Keyboard

The BBC Micro has a smart appearance. Its cream coloured plastic case is reasonably sturdy and its keyboard has a neat layout. Above the standard querty keyboard there are nine red programmable 'function' keys, which can be 'defined' to generate any character or string. There are four cursor control keys at the top right of the keyboard and these and the COPY keys are a lighter brown than the rest of the keys. The copy facility is one that I really appreciated, both when writing or altering programs or when text processing. One criticism of the keyboard is that the spacebar responds only if hit near the middle, and it is possible to type a whole line without ever throwing a space!

Hardware

It's quite easy to lift the lid off the BBC Micro – you simply remove four screws. The layout inside is neat. There is just one PCB and all the chips are neatly socketed. The 2MHz 6502 is in the centre of the board. If you are aware of the early history of the BBC project you may know that the 6502 was specifically ruled out in the early specifications for the machine, which stipulated a Z80 CPU. However Acorn already used the 6502 in their ATOM and had developed a BASIC interpreter, so it obviously made sense to stick with the 6502. This decision certainly paid off and the result, BBC BASIC, is one of the micro's winning points.



BBC BASIC

The great thing about BBC BASIC is that it has all the advantages of BASIC (simplicity) plus the advantages of a structured language such as Pascal. Using its 'procedures and functions' you can build up large programs from smaller routines. Once you've got used to this method of programming life is much easier, especially when it comes to debugging, making modifications, or adding extra routines.

BBC BASIC is fast – to my knowledge it is the fastest BASIC running on a popular micro – and it is easy to mix assembler and BASIC, which means you can speed up critical parts of your programs even further.

Sound and Graphics

One of the most versatile features of the BBC Micro is its graphics displays. The Model B machine can operate in any of eight modes (but only three are available on the Model A

Continued on Page 53

machine). It is possible to use a maximum of 16 colours, of which 8 are solid colours and 8 flashing combinations. Unlike some other computers the colours are clear and true and can be displayed anywhere on the screen.

The sound is also a noteworthy feature. There are three tone channels and they can be used simultaneously to produce three note chords and so on. As well as the SOUND command, which can be used to produce 'pure' tones, there is the ENVELOPE command which can be used to modulate the quality of the sound.

Software

Acorn's in-house software, Acornsoft, now has an impressive list of cassette titles including games, educational packages and two alternative languages, LISP and FORTH. Some of these will, of course, only run on the Model B. A wide range of programmes are also available from independant software houses.

Expansion

The BBC Micro (Model B) is not just another all-singing, all-dancing home computer; it is the basis of a really powerful system and one that will last into an exciting future. The Model B (only) has an RS423 serial interface, an 8-bit centronics interface (with a 26-pin ribbon cable plug) and an 8-bit user port (which is used via a 20-pin ribbon cable plug).

The printer port, the user port and the disc drive port are to be found on the underside of the Model B's case. Next to them are two other interfaces. The first is provided for connection to (among other things) Teletext and Prestel, and the second is provided so that the Micro can be equipped with a second processor. The BBC Micro is therefore ready for the 'telesoftware developments of the future.

Both Models A and B can be fitted with an Econet interface, and therefore connected in a large number to form a Local Area Network.

In its price range the potential of the BBC Model B is enormous, and as far as I am concerned the BBC Model B Micro takes first prize in almost any competition.

Commodore VIC-20

The VIC-20 is the predecessor of the Commodore 64. It has just 5K of standard memory, expandable to 32K; and for this reason it is being rapidly overtaken by newer, cheaper and equally powerful machines with greater storage capacity.

Keyboard

One of the main plus points of the VIC is its high quality typewriter style keyboard, laid out in standard qwerty fashion with special purpose keys at the edges. A shift key is present on both sides with a lock on the left hand side.

A lot of thought has been put into the position of the symbol keys, especially those used frequently when programming; many are non-shifted and this is a real time saver. The cassette unit, the Commodore C2N, perfectly matches the styling of the VIC-20 being of a similar robust construction and identical colouring.

Sounds and Graphics

The music and colours, although flexible and extensive, are disappointing. When used in a program these must be controlled by POKEs. To a beginner this will be rather confusing and does tend to be wasteful of memory. Sophisticated subroutines using parameters can be used but these need a fair amount of experience to develop.

The music scale covers a range of 128 notes each of which can be played through any combination of three speakers. A fourth 'white noise' speaker, for explosions etc., is useable.

Documentation

The VIC-20 manuals, especially the introduction to BASIC, are excellent. The course covers the VIC and BASIC in a logical manner and is easy to follow, if a little verbose in places. Plenty of practice at the basic skills is provided, with various exercises and cassette programs for backup. there is even a speed typing routine which provides a dynamic grading.

Acorn Electron

The Electron is best summed up as a 'stripped down' BBC Micro which runs at a slower speed (by a factor varying between 1.5 and 3, depending upon what it is doing). The Electron is stripped down in the sense that it has all of the features of the BBC Micro apart from teletext graphics mode, a second cassette speed, space for extra ROMs and the full range of interfaces. Specifically, it lacks a serial interface, joystick interface, 1MHz and Tube interfaces and the printer and user ports. It will be possible to add all of the missing features apart from teletext graphics and 300 baud cassette loading. However, the question really is why would anyone prefer to buy the Electron rather than a BBC Micro?

The most obvious answer is price. The Electron provides a low cost route into the world of 'BBC computing'. In other words, with the Electron you can take advantage of all the software, the books and the media razamatazz that has been attendant on the BBC's Computer Literacy Project. Although you will have the possibility of up-grading to a fuller system as finances allow, the Electron is a more than adequate computer as it stands. It has a specification that makes it suitable for use as a games computer, an educational computer, a graphics computer, a low cost development system . . . and, by adding only one or two of the promised expansion modules its range is likely to be even greater.

The Electron doesn't entirely fill the role currently occupied by the BBC Micro but there are many applications that the Electron can be used in at lower cost and without a noticeable loss of performance.

Continued on Page 54

Commodore 64

The Commodore 64 is a machine that is being offered to two sections of the market: to the games playing community it is presented as an improved VIC 20, and to the business community it is presented as a smaller, cheaper PET or Series 500 machine. If you look at the characteristics of the Commodore 64 then you can see how the confusion arises. With a 40 column colour display and good sound facilities it should be good for games. However if you add a Z80 card and CP/M to it then you have a candidate for more serious tasks.

Hardware

This machine is superficially like the VIC 20. It certainly fits inside a plastic case that is the same size as the VIC 20's and uses the same keyboard, but its overall specifications are much better. The only external detail worth commenting on is the use of a separate power supply.

Commodore owns the chip manufacturing company MOS Technology, and any machines it makes can and usually does include custom designed chips, and so it is with the Commodore 64. Inside you will find a special version of the well known 6502 microprocessor designated the 6510, two video processor chips and a sound generator chip.

As implied in its name, this machine packs a full 64K of RAM. However while running BASIC only about 38K of RAM is available; the rest of the memory space is taken up by 20K of BASIC ROM. If you don't need BASIC then the ROM can be switched out and more of the RAM used. Apart from this ability to reconfigure the memory the 6510 seems to be a fairly standard 6502 and capable of running 6502 machine code without any modifications.

Software

The Commodore 64 uses the early version of Microsoft BASIC that has become associated with the PET and for this reason is often known as PET BASIC. To be more precise it is the same version of BASIC as used on the VIC 20. There is a lot to be said for sticking with a well-known version of BASIC but even the Microsoft variety has moved on since the PET and now includes extra features that makes it more attractive. Seen against this background the BASIC on the 64 is primitive. What is even worse is that no extra facilities to control the unusual hardware are included.

Expansion

Cassette is usually cheaper than disc for program storage, but in the Commodore 64's case you have to use one of the specially made tape drives costing around fifty pounds. To fit its image as a games machine the 64 has two joystick ports. However, in line with its business image you can buy an IEEE 488 interface cartridge that will give you access to the wide range of PET peripherals, and a Z80 card that will allow you to use CP/M and a wide range of business software.

Documentation

Commodore have a reputation for producing incomprehensible computer manuals, but the 64's documentation is well produced and fairly readable. It is aimed at the beginner and so lacks any real technical information. For the more technical minded programmer there is a more advanced manual available (at extra cost).

In Conclusion

The Commodore 64 is a machine with a great deal of potential unfortunately limited by its 40 column screen. Perhaps the best description is as an expensive games machine that offers the possibility of some serious use. Without a good extended BASIC the complete novice will be better off using package software or not using all of the advanced hardware within the machine. In short, the Commodore 64 is a machine in search of a good language.

Dragon 32

The first thing to say about the Dragon is that from the point of view of its electronics and its BASIC it is essentially little different from the Tandy Color Computer. Indeed the BASIC used by both machines is Microsoft Color BASIC and so programs written for one should run on the other.

Keyboard

The Dragon is astonishingly light – it weighs just four and a half pounds – and continues the contemporary trend for having part of the power supply in a separate box. Overall, the case is well styled, though the plastic is a little on the thin and brittle side. The keyboard is a full-function typewriter style unit. The text screen is 16 lines by 32 characters and lower case characters display as upper case in reversed colour. The small screen display can be forgiven in a machine of this price but the lack of any real lower case characters is a serious omission.

Graphics

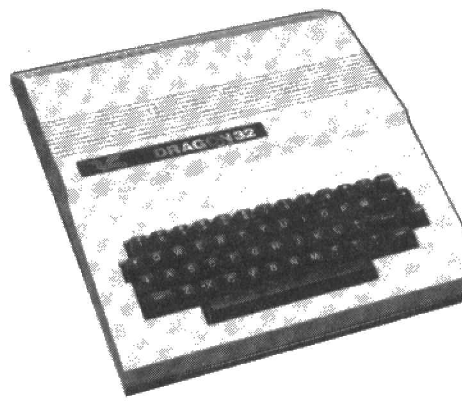
There are two types of screen display directly available from BASIC, a text and low resolution (64 by 32) eight colour screen and five high resolution screens. The low resolution graphics are obtained by printing 2 by 2 block graphics characters on the text screen. You can plot or remove a point of any of the eight colours using the Tandy style SET and RESET commands.

Sound

The sound generator in the Dragon is simpler than most. It can produce up to 255 separate tones, but only one at a time (from BASIC). In other words you cannot produce chords of any kind. This might be a disadvantage if you are particularly interested in music but it is good enough for playing games. On the plus side the BASIC commands that control the sound generator are particularly easy to use and mean that although the sound generator isn't as sophisticated as some it will at least be used!

Hardware

The microprocessor inside the Dragon is an unusual one and worthy of mention: it is the Motorola 6809. This is an advanced microprocessor somewhere between an eight bit and sixteen bit design. There is no doubt that it is one of the best microprocessors around



and the Dragon is all the better for using it. However, it does cause some problems if you want to use the machine code and are used to the Z80 or the 6502.

The 32K of RAM is in the form of eight 4864 chips and there is no space for on-board memory expansion but you shouldn't need any either! The BASIC is in two 8K ROMs and there is room in the address map for an additional 16K of ROM in the form of a cartridge.

Software

The Dragon comes equipped with what Tandy would call 'Extended Color BASIC'. Even though the Dragon uses the 6809 rather than a Z80 the BASIC that it runs is essentially the Microsoft BASIC that is so common on other machines. You will find all the usual statements: IF . . . THEN . . . ELSE, PRINT USING, INPUT LINE etc. Also strings are handled using the functions LEFT$, RIGHT$ and MID$. So, learning Dragon BASIC will stand you in good stead for using other machines.

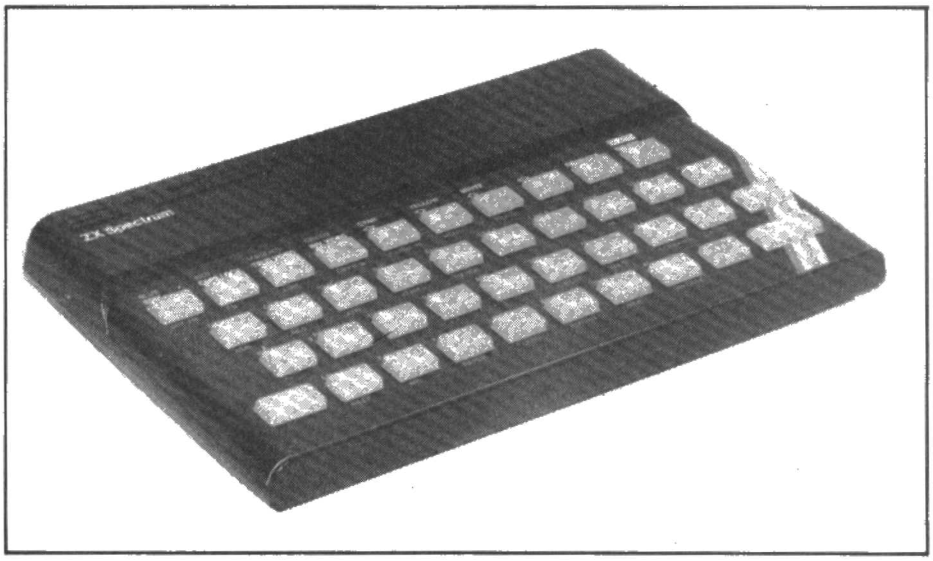
Sinclair Spectrum

The first thing which a prospective buyer would notice about the Sinclair (any Sinclair) if they took it home for a week, is that there is only one thing to learn. On all other computers, you have to learn the language, and if you are not already familiar with typewriters, you have to learn the keyboard. On this machine the language is the keyboard.

In other words, instead of typing PRINT, you press the P key and the machine types PRINT for you.

This sound wonderful until you realise that there are around 90 different words in the BASIC language. There are 40 keys on the Sinclair keyboard, and they also have to carry the number 0 to 9, the letters A to Z, all the punctuation, calculation, and editing functions, plus colour codes. At that point only the fact that you have spent all your money on the machine persuades you to stick at it and learn the way around the keyboard.

But the advantages go deep, for all their apparent elusiveness. On a Sinclair you are never suddenly stuck for a command, because it's there in front of you. Nine times out of ten, if you do use the command incorrectly, the machine tells you exactly where you have gone wrong.



Spectrum BASIC

Spectrum BASIC is a noticeable improvement on the ZX81 version. I don't think the speed of BASIC matters too much to most people, which is just as well for the Spectrum. The only thing slower is a Casio pocket calculator, or the ZXC81 in SLOW mode. In FAST mode the ZX81 beats the Spectrum comfortably for speed, and any model BBC micro is between four and ten times faster.

Graphics

In high-resolution graphics there are point, line, circle and arc drawing commands. There are 16 pre-defined graphics characters plus 21 user-definable graphics characters. Also available are functions to yield character at a given position, attribute at a given position (colours, brightness and flash) and whether a given pixel is set. Text may be written on the screen on 24 lines of 32 characters, and text and graphics may be freely mixed.

Features

Features of the Spectrum can be placed into 7 categories.

- The first is unique to the Spectrum: the Sinclair Microdrive. The Microdrive is a memory storage device (up to 100K) which uses cartridges of continuous loop tape. Access time is 3.5 seconds and therefore far quicker than conventional cassette tape. Together with an interface for connection to the computer, the Microdrive costs £80, and therefore bridges the price gap between cassette and disk storage.

- The Spectrum uses an enhanced BASIC language. It still omits the one thing which I think essential in such a machine, and that is easy access to machine code, but for all its faults, Sinclair BASIC is useable and widely known.

- The Spectrum is a colour machine with programmable graphics.

- The Keyboard has its faults, but is generally quite adequate. At least you can use it without wondering if your finger has slipped off the key.

- Sound generation. There is a silly design fault here, and it is one which I cannot forgive: the sound is inaudibly soft, and it takes a tedious performance of plugging and unplugging to amplify it.

- The Spectrum has a facility which was sadly missed on the ZX81: an operating system. In that machine, only program save and load were available, whereas in the Spectrum we get screen handling, tape handling, terminals handling and network handling.

- Finally, the Spectrum keys have an auto-repeat facility. Perhaps this should have come at the top of the list, because it is something I'd pay an extra £25 for on the ZX81, with no questions asked.

Sinclair ZX81

The machine that led the low-price home computer boom really needs no introduction. The ZX81 is simple and unsophisticated. The standard machine has a tiny 1K of memory expandable to 16K (and further), and the keyboard is of the membrane type; rather difficult to use, but all BASIC commands can be entered at the touch of a single key, as on the Spectrum.

On the negative side, the machine is very small and light, and doesn't take kindly to the fat fingered or cack-handed. There is also a design fault in the connection from the computer to the 16K RAM pack, which wobbles, sometimes resulting in the loss of a painstakingly entered program. The ZX81 graphics are black and white only, and created with a selection of 16 pre-defined characters.

To summarise, the ZX81 is the ideal beginners machine. It is cheap, robust, and simple to use. A wide variety of software is available, and the standard machine can be upgraded using the many peripheral and extension devices sold on the market. Whilst the handbook could be improved upon in certain sections, the use of single-key commands greatly simplifies the process of learning Sinclair's idiosyncratic version of BASIC, and the principles of programming in general.

Oric 1

Keyboard

The Oric is a rather smart looking machine in a plastic case that measures just seven by eleven inches. A novel feature is the use of a wedge shaped case to give a convenient angle to the keyboard. The keys are made of hard plastic and move but they can hardly be compared to the switches on a real keyboard. This is not to say that the Oric's keyboard isn't comprehensive: it has every key you could possibly want and a few more, but it somehow feels slow. There are no Spectrum style keywords and each command must be entered letter by letter.

The standard text screen is composed of 28 rows of 40 characters with both upper and lower case. The screen scrolls automatically when full and all in all it gives the feel of a full sized computer. In this mode the Oric can be programmed in a fairly standard form of Microsoft BASIC and this is one of its strong points. If you specifically want a computer to develop Microsoft programs then the Oric is a bargain! However, although there is the facility to save and load programs on tape there appears to be no way of saving data on tape, so developing data processing programs might be a little difficult.

Graphics

Good graphics are a must for any small home computer, and the Oric's are good enough for playing games and many serious applications. If you need a low cost Teletext or Prestel graphics display then the Oric might well be just the machine for you. The Oric has two main display modes: text/lo-res and high-res graphics. The low resolution screen is 39 wide by 27 high but the effective resolution is higher, in that you can use teletext graphics blocks within each graphics cell. In this mode you can also mix text and graphics in eight different colours.

The high resolution graphics mode provides 240 x 200 plotting points in any two of the eight colours. If you want to use all eight colours then you are reduced to an effective colour resolution of 40 blocks by 200 lines.

Sound

The sound capabilities of the Oric are surpassed only by the BBC Micro which is, of course, a more expensive machine altogether. There are a number of predefined noises such as ZAP, EXPLODE etc. that are just right for writing games programs. There is a MUSIC command to help you translate music to play on your Oric and last but not least a PLAY command that provides a limited but very easy to use envelope facility to make non-standard special effects.

Hardware

The hardware design and construction of the Oric is very good. It is nice to see that it is

Documentation

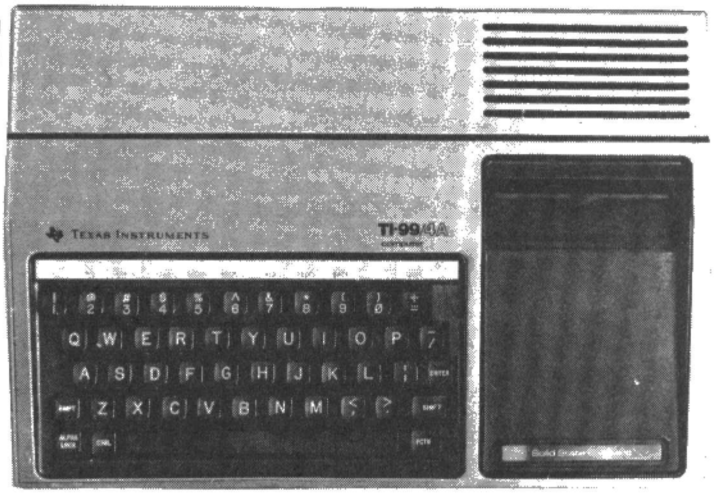
Well written and fine for learning BASIC and the fundamentals of the Oric, but there is a real need for another more advanced manual to help users get the best from their machine and use it in a really creative way.

possible to do clever things with a 6502 instead of the usual Z80! However, with the presence of the 6522 and the sound effects chip it is surprising that a few extras such as a joystick input and timer are missing.

Expansion

One of the attractive things about the Oric is the range of expansion that is available or promised, including a modem interface, disc unit and special printer. The Oric, with its teletext/prestel graphics and a modem opens up new areas in affordable computing. If there is a particular feature that you feel is important for your application, e.g. its keyboard or Microsoft BASIC, then it is a machine to consider carefully.

Continued on Page 62



TI-99/4A

A major feature of the TI-99/4 is the use of plug in software modules. If you want to use a particular program all you have to do is push the appropriate program module into the slot at the front of the machine and switch on. The first thing that you see on the screen is the 'startup frame' announcing 'Texas Instruments Home Computer' and showing some very pretty colour bars. If you press any key you immediately jump to a menu asking you to select what you want to do – run TI BASIC or one of the programs contained in the software module. If you select one of the programs in the software module the response is instantaneous – you don't have to wait for a cassette to load or fiddle with volume controls etc. This is obviously an advantage if you want to use a program with the minimum of fuss and is ideal for educational use. If you want to save and load your own BASIC programs however then you will have to get involved with more wires and a cassette unit (not supplied).

Graphics

The TV display is 32 characters by 24 lines of very clear characters. This is of course well below the number of characters and lines you would expect from a standard business machine (i.e. 80 characters by 24 lines) but is OK for a home computer. The stunning graphics that so many of the software modules produce are generated not by a separate graphics facility but simply by making use of the fact that you can define your own characters sets. As each character is made from 8 by 8 dots this gives you an effective resolution of 256 by 192. If you use a colour TV set then you will be rewarded by 16 colours that are used by most of the software modules to very good effect. If you only use a black and white set then you will be able to display a reasonable range of tones – black through grey to white.

Sound

Coupled with the good colour graphics is an excellent sound generator. It is capable of three simultaneous tones over three octaves plus a noise generator for the pops and bangs that are an essential part of computer games. All of the sound (including that from the optional speech synthesiser) is produced through the TV set speaker.

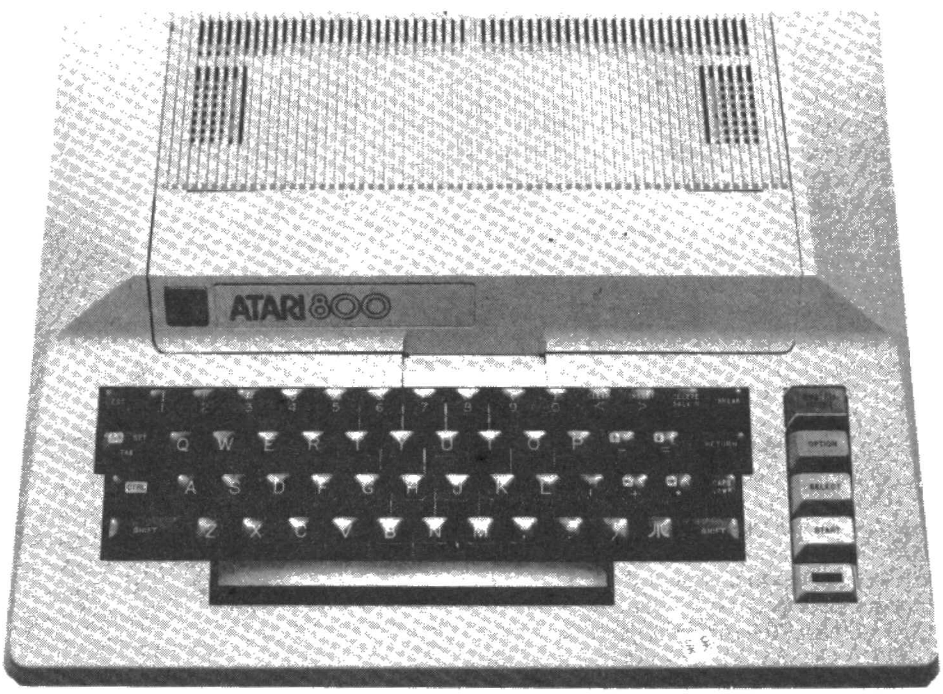
Documentation

In contrast to most personal computers the TI-99/4A comes with a good beginners' manual and a reasonable advanced manual. If you find these two insufficient then you can also buy a specially written book "Programming BASIC with the TI Home Computer" by Herbert Peckman.

Conclusion

The TI-99/4A is good value for money and makes an excellent home computer. The features of this machine are tailor made for good animated graphics, speech synthesis and game playing. Once you've acquired the machine you can hang on the peripherals to build it into one which has the capacity, though unfortunately not the speed, of lots of systems that come in much bigger cases. The TI-99/4A is a computer that the whole family can have fun with. It will appeal to every generation and is a particularly suitable choice for introducing children to computers.

Atari 400 Atari 800



Both these machines, the Atari 400 and Atari 800, were originally designed primarily as games machines. They are however worthy of serious consideration, and recently Atari have started to lower their prices considerably and sell them as general purpose microcomputers at the heart of an expandable and logical system. In addition there is much to be learned about hardware techniques from both models.

Keyboard

The substantial differences between the two machines are firstly in memory storage (16K on the 400 and 48K on the 800) and in the keyboards: the 400 keyboard is a flat membrane type whereas the 800 has a standard quality typewriter keyboard. The membrane keyboard is slow and tiring to use and it is probably worth paying the substantial price difference to obtain the benefits of the typewriter keyboard. Otherwise, there is so little difference between these two machines that for the rest of this review they will be treated as one.

Graphics

The Atari's CPU is a 1.8MHz 6502, fast enough to be considered double-speed and this is reflected in the machine's performance. The graphics and sound quality are very good indeed. There is a problem in that the Atari lacks an 80 column text mode, but a very clear upper/lower case 40 column display compensates for this omission. A feature of the Atari graphics is the way that the display colours can be selected from a very wide range of colours (256) specified in terms of colour and brightness. The colour of any area of the screen isn't defined directly by the code stored in the screen memory; instead the code selects which of a number of 'colour registers' holds the colour information. This, in Atari jargon, is 'colour indirection', and very useful it is too. You can plot areas of the screen and change their colour instantaneously by changing what is stored in the colour register.

Language

Atari BASIC is a good dialect, not modern, not too old fashioned, but it fails to tackle the problem of making the exceptionally good hardware facilities of the Atari accessible to the user. Moreover, it tends to make the facilities that it does offer difficult to use. I cannot help but feel that Atari BASIC strangles an otherwise excellent machine. On the plus side, the program editing facilities are very easy to use and very logical. A wide variety of alternative languages are available as extras in program packs, including Pascal, Forth, and another version of Microsoft BASIC.

Expansion

The Atari builds into a complete computer system not by plugging in extra cards but by plugging in separate units. The only two expansion units are the modem/communication module and the disc drive module. The modem/comms module is very good value, including four serial ports and one parallel printer port. This module can be used not only to drive a printer but also for taking part in computer networks for transferring information and programs.

The disc drive module adds a five inch disc to the system. As this is a specially constructed drive it is more expensive than standard units, and it is also slow in operation. The disc operating system that the Atari uses is adequate from the technical point of view and has a considerable advantage in being easy to use via an extensive menu.

Documentation

The Atari's BASIC documentation is very poor. It is issued as a loose leaf collection of paper that is intended for a three hole ring binder (but not supplied with one!) Description of how commands work are generally insufficient and sometimes incomplete. Perhaps the most exciting way of using the Atari is from assembler and this is likely to be the way that *E&CM* readers get the most from the machine. While Atari issues very little technical information for the assembly language programmer there are a number of very good books on both Atari assembler and the inner workings of the Atari, so you should be able to find out all you need to know – at a price.

Conclusion

The Atari is a very powerful and versatile machine. It is clearly well designed and very advanced. In my opinion it has one or two drawbacks with respect to its graphics display that are due to the designers not really thinking about the way that people use colour and shape in displays. For example, the single colour character against a fixed background colour limits the way that user defined graphics can be put together. However these shortcomings are more than made up for in some of the more unusual features that the video display offers – sprites, dynamic character definition, smooth scrolling, 256 colours, colour indirection, etc., etc.

For anyone considering buying either Atari model the single biggest factor to take into account is the very poor version of BASIC that has come to be the accepted standard. Atari BASIC succeeds in reducing a good hardware design into a restricted and awkward machine. It is one of the best examples that I have come across of software working against hardware. However, it is not too late for Atari to realise how important BASIC is for a personal computer, and either find a better version to supply (free!) with the machine, or even to write one of their own; this would give the Atari a new lease of life and allow it to compete with the newer machines on level terms.

FORTH BUYER'S GUIDE

Nick Clare looks at some of the packages that endow a range of popular micros with the ability to run FORTH programs.

In the September 1982 issue of *E&CM* Nigel Freestone began his article on FORTH computers by stating that FORTH is flexible, interactive, compact, fast (very fast), transportable and that FORTH is not new. Readers wanting to know more about the ins and outs of FORTH should refer back to this article and subsequent parts as the purpose of this feature is to briefly look at the availability of FORTH packages rather than at the language itself.

Sinclair FORTH

Sinclair market cassette based FORTH packages for both the ZX81 and Spectrum under their own name although these have both been developed by the well known company, Artic Computing.

The ZX81 version is designed for the 16K machine and comes complete with two manuals (one a User's manual and the other describing the operation of the editor) as well as a new keyboard overlay.

The FORTH compiler takes about six minutes to load whereupon the language will introduce itself and allow programming to begin. Loading the editor requires that

1 LOAD N/L

is entered and a total of four editor screens loaded (each takes about 20 seconds).

Both manuals, although brief, are well written and the programs all loaded without any problems.

Sinclair's/Artic's package for the Spectrum is very much the same as that for the ZX81 (a 48K is required to run the compiler) and both provide a very useful introduction to FORTH without the need to invest in any additional hardware.

Abersoft also produce a version of FORTH designed for the 48K Spectrum and the manual for this extended fig-FORTH package begins with the statement that it is not intended to be a tutorial on the language. The Abersoft package does however contain some additions to the standard and these are all explained with examples.

The computer is loaded by one LOAD command and when the copyright message and version number appears on the screen, it is only necessary to hit return to begin programming in FORTH.

BBC 4

While all the FORTH packages so far described have been cassette based versions that can require an appreciable time to load, the HCCS implementation for the BBC micro is supplied as an EPROM that can be installed in one of the machines sideways ROM sockets.

The compiler is suitable for all versions of the operating system having recently been modified to suit OS1.0 and above. Full installation instructions for all versions of the machine are provided along with a plea that Mrrs. Acorn finally decide on what the last operating system for the machine is going to be.

Depending on how the EPROM is installed, FORTH will either be available as the default language or will be ready after a *FORTH call.

Documentation is comprehensive and includes a glossary of the FORTH nucleus dictionary and a full description of the extra FORTH words that have been included to take advantage of the facilities provided by the BBC micro.

Dragon FORTH

M & J's fig-FORTH compiler for the Dragon also provides a very useful 6809 micro assembler. The cassette based implementation comes complete with two very comprehensive manuals.

Versions are available for both the 32K and 64K variants of the Dragon and, in addition the company are about to launch a disk based version of the package.

A major new feature recently introduced to M & J's package is the ability to use BASIC commands in the FORTH environment.

Learning About FORTH

All the manuals that accompany the various FORTH compilers described above stress that they are not intended to be tutorials in the language. In the main they recommend that one of the many books dealing with FORTH are consulted.

To this end M & J offer Haydon's 'All About FORTH' at £7.95 (much cheaper than the USA import price) while other titles that receive recommendation are, for beginners, 'Starting FORTH' by Leo Brodie and for the more advanced user, C. H. Ting's 'The Systems Guide To fig-FORTH'.

Sinclair Research
Freepost
Camberley
Surrey
GU15 3BR

Abersoft
7 Maes Afallen
Bow Street
Dyfed
SY24 5BA

ACCS Associates
533 Durham Road
Low Fell
Gateshead
Tyne and Wear
NE9 5EY

M & J Software
34 Grays Close
Schular Green
Stoke on Trent
ST7 3LU

MICRO GRAPHICS TECHNIQUES

In last month's Micro Graphics the fundamental idea that lies behind computer animation – the sprite – was introduced. This month Mike James explores further the problems of internal animation and controlling motion.

The Structure of Animation

The sprite, by which we mean a user-defined shape associated with a position, a velocity and a rule for determining its acceleration, is the fundamental data type used in computer animation. In this sense the sprite plays the role in animation programs that the string does in text handling and the array does in numerical programming. Not only does the sprite provide the fundamental data for animation it also presents a simple organisation for animation programs.

The flow diagram in **Figure 1** shows the ideal organisation of an animation program. The variable T is the animation counter in the sense that its value indicates how many sprite updates have occurred. In many applications the value in T can be treated as a measure of the time that the program has been running. The core of the animation loop consists of just three main procedures – a routine to remove all the sprites from the screen, a routine to update the sprite positions etc., and a routine to place the sprite back on the screen in their new position. Before this loop is repeated again a routine is used to check for any sprite events – collisions etc. – that might have occurred. If an event is detected then a special part of the program is called that handles the event. What happens after this handling depends very much on its nature. Some events signal the end of the program and others cause the animation loop to be restarted.

The only trouble with this ideal animation loop is that it assumes that the time between blanking all the sprites and reprinting them is very short. In fact if this time is much longer than one or two TV frame periods the user begins to see the screen going blank as all the sprites are removed for updating. Of course if you are using BASIC then this is almost certain to be the case and a modification to this fast animation method is called for. The slow animation loop shown in **Figure 2** can be used to produce flicker free animation no matter how long it takes to go round the loop. The only difference between the two methods is that instead of applying each routine – blanking, updating and reprinting – to all of the sprites, each sprite is dealt with in turn. Also to minimise the time between blanking and reprinting the updating is carried out first. Notice that it is important to check for any sprite events at the END of the animation loop. The reason for this is that it is essential to check for sprite events on the basis of the screen as it appears to the user. For example, if you checked for a collision between two sprites during the animation loop the chances are that you would incorrectly detect a collision between the new position of one sprite and the old position of the other!

The fast animation loop is the simpler of the two because it allows each of the routines to be written independently, whereas the slow animation routine requires rather more information to be passed to each routine. However, for BASIC programs the slow animation loop is about the only way to do things and it is the method used by the animation program given last month.

Both methods share two important features: the animation counter T and the sprite event detection routine, and these form the subject of the rest of this article.

Internal Animation

One of the most impressive features of 'Space Invaders' on the Atari computer is the way that each of the invaders 'waves its arms' as they move from side to side. So far all the sprites that have been described have kept their shape fixed as they moved. However the internal animation, exemplified by Atari Space Invaders is not difficult to achieve, and what is more surprising adds very little to the complexity or time taken to animate a sprite.

The key to internal animation is to replace the string that is used to hold the single shape normally associated with a sprite, by a string or character array that holds a range of shapes. The shapes stored in the array form a sequence that will produce the desired internal animation. As the sprite moves around the screen, the animation counter T is used to generate an index governing which of the shapes will be printed on the screen by the animation loop. Perhaps the easiest way to explain this idea is by way of an example.

The following program will animate a little man shaped figure as it falls toward the bottom of the screen. The internal animation takes the

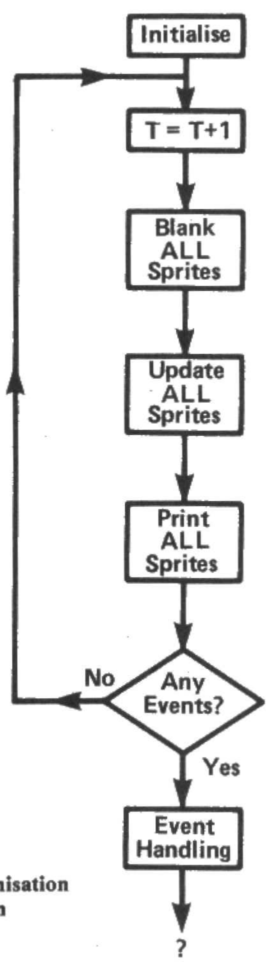


Figure 1. The ideal organisation of an animation program.

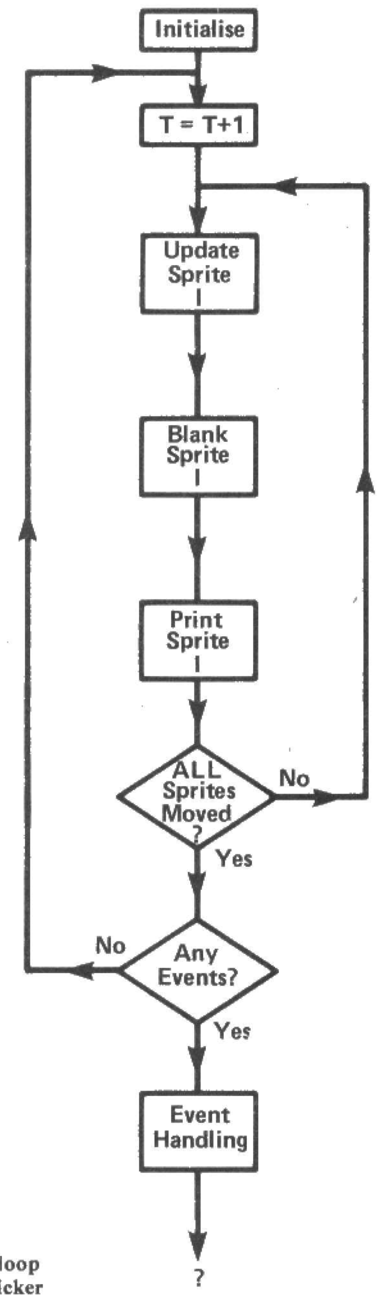


Figure 2.
A slow animation loop used to produce flicker free animation.

form of making the man flap his arms as if in a vain attempt to fly back up the screen. The sequence of shapes necessary to implement this internal animation can be seen in **Figure 3.** These can be used to define three user-defined characters on the BBC Micro or Spectrum. Oric owners will have to modify the shapes to fit into their 6 by 8 grid and the Dragon owners will have to use GET and PUT.

The full animation sequence involves four different stages. First shape 0 is used, then shape 1 and shape 2, followed by a re-use of shape 1 again before the whole sequence repeats. To make things simpler it is better to consider the re-use of shape 1 a separate step in the animation sequence and allocate it an additional array element. Thus if the shapes are stored in the array S$(3) the animation sequence is S$(0), S$(1), S$(2), S$(3), S$(0) and so on where S$(1) and S$(3) both contain the same user-defined shape. Using this arrangement the animation counter can easily be used to select the correct shape during the animation. In other words if the animation counter starts from 0 we have:

T =	0	1	2	3	4	5	6	7
Use	S$(0)	S$(1)	S$(2)	S$(3)	S$(0)	S$(1)	S$(2)	S$(3)

If you look at this pattern carefully you should be able to see that the index of the array is simply the remainder when T is divided by 4. This is exactly the method used to select which shape should be displayed using the animation counter. If the internal animation sequence consists of N shapes then the index is simply the remainder after dividing the animation counter by N.

Returning to the flapping falling man program, this gives for the BBC Micro:

```
10 GOSUB 1000
20 GOSUB 2000
30 T=-1
40 T=T+1
45 FOR J=1 TO 100:NEXT J
50 GOSUB 3000
60 GOSUB 4000
70 GOTO 40

1000 MODE 5
1010 RETURN

2000 VDU 23,224,&99,&5A,&3C,&18,&18,&24,&24,&24
2010 VDU 23,225,&18,&18,&FF,&18,&18,&24,&24,&24
2020 VDU 23,226,&18,&18,&3C,&5A,&99,&24,&24,&24
2030 DIM S$(3)
2040 S$(0)=CHR$(224)
2050 S$(1)=CHR$(225)
2060 S$(2)=CHR$(226)
2070 S$(3)=CHR$(225)
2080 X=10
2090 Y=3
2100 XV=0
2110 YV=0
2120 XA=0
2130 YA=.1
2140 RETURN

3000 XP=X:YP=Y
3010 X=X+XV
3020 Y=Y+YV
3030 XV=XV+XA
3040 YV=YV+YA
3050 IF Y>20 THEN YA=-YV
3060 RETURN

4000 PRINT TAB(XP,YP);" ";
4010 R=T-INT(T/4)*4
4020 PRINT TAB(X,Y);S$(R);
4030 RETURN
```

The program should be easy to convert to the Spectrum. Subroutine 1000 initialises the machine and subroutine 2000 initialises the single sprite that this program uses. As there is only one sprite there is no need to use arrays to store position, velocity and acceleration. Subroutine 3000 performs the usual sprite updates and subroutine 4000 blanks out and re-prints the sprite. The only real difference between this program and last month's sprite program is the way that the animation counter is used to select which shape will be printed.

Notice that as the flying man only moves vertically there is really no need to carry out any updates on the X position velocity etc. However, to make the program completely general (if a little slower than it need be) these unnecessary updates are included. You can introduce internal animation to any number of sprites in exactly the same way subject, of course, to the condition that you can move everything fast enough.

Sprite Events

The idea that lies behind sprite events was described in last month's Micro Graphics, but in short a sprite event is any detectable condition that requires something other than the standard animation sequence to handle it. The best way to illustrate this idea is to add a simple event detector and handler to the flying man program given earlier. At the moment when the man finally reaches the end of his fall nothing exciting happens. To rectify this simply entails two extra subroutines.

```
65 GOSUB 5000

5000 IF Y<=20 THEN RETURN
5010 GOSUB 6000
5020 RETURN
```

```
6000 SOUND 0,-15,4,20
6010 END
```

Subroutine 5000 checks if the man has reached the ground or not and transfers control to the event handler 6000 if he has. In this case subroutine 6000 only makes an explosive noise and then stops the program but in principle it could be a much more complicated routine than this.

As a more complicated example the following modifications to last month's multiple sprite program will add a crash noise each time a collision occurs and remove the pair of sprites involved.

```
  85 GOSUB 7000

7000 REM Check for collisions
7010 FOR I=1 TO N
7015 IF S$(I)="" THEN GOTO 7060
7020 FOR J=1 TO N
7030 IF I=J THEN GOTO 7050
7040 IF X(I)=X(J) AND Y(I)=Y(J) THEN GOSUB 8000
7050 NEXT J
7060 NEXT I
7070 RETURN

8000 S$(I)=""
8010 S$(J)=""
8020 SOUND 0,-15,4,10
8030 RETURN
```

The above lines should be added to the first version of the multiple sprite program given last month. The collision check routine, subroutine 7000, is not the most efficient possible as it checks each part of sprites twice but it is easy to understand.

Explosions – Terminal Events!

The most common use of sprite events is to detect a condition that implies the end of a game. For example, if a missile hits its target then the sprite collision involved would signal the end of the game. It is surprising how often end of game sequences involve animated explosions. It is possible to write pages on how sprites can meet their final end in spectacular explosions but the main trouble is that most of them need the speed of assembler to be effective. The simplest and quickest method to produce an explosion is to define an explosion character that can be printed over the current sprites position. To define a good explosion character is not difficult but you should try to avoid using too many of the pixels near the edge of the character so that the outline of the 8 by 8 character square is invisible.

More advanced methods of constructing explosions are nearly all based on either making the explosion appear to grow by printing a sequence of explosion characters or on using high resolution graphics commands to make the pixels of the shape appear to move apart. As already mentioned to do this smoothly and convincingly needs assembler and indeed a good explosion routine can take an unreasonable amount of code to produce! However, you might like to try either of the following two explosion routines. The first makes use of a short sequence of expanding explosion characters —

```
 10 MODE 5
 20 VDU 23,224,0,0,0,&01,&08,0,0,0
 30 VDU 23,225,0,0,&04,&18,&18,&14,0,0
 40 VDU 23,226,0,&04,&24,&1C,&72,&14,&24,0
 50 VDU 23,227,&82,&24,&3C,&1C,&FC,&7A,&48,&85
 60 PRINT TAB(10,10);"";
 70 FOR I=224 TO 227
 80 PRINT TAB (10,10);CHR$(I);
 90 FOR J=1 TO 200
100 NEXT J
110 NEXT I
120 FOR J=1 TO 1000:NEXT J
130 GOTO 60
```

and the second simply plots radial lines from the centre of the character

```
10 MODE 5
20 X=500
30 Y=500
40 DX=50-RND(100)
50 DY=50-RND(100)
60 MOVE X,Y
70 PLOT 1,DX,DY
80 GOTO 40
```

Notice that to use the above routine in a low resolution graphics program involves converting low res co-ordinates to high res co-ordinates. In general explosion routines are very much improved by good sound effects and this makes sound an even more important component of BASIC explosions.

Conclusion

By this point you should be able to see the strengths of the sprite approach to animation and also start to see the difficulties. In an ideal world with super fast computers then it would be possible to apply the sprite approach to all animation tasks. Even the tricky problem of producing explosions could be discussed in terms of pixel sized sprites if only computer power would make it practical. Perhaps an ideal solution would be to use special sprite hardware to speed things up. Sprite hardware tends to offer only a limited and restricted number of facilities. There is a lot to be said for an assembly language sprite package that the BASIC programmer could gain access to.

Next month – The subject of sprites is concluded with an examination of keyboard control of their movement, and, still in the field of animation, Mike James studies techniques of building backgrounds and animating large objects.

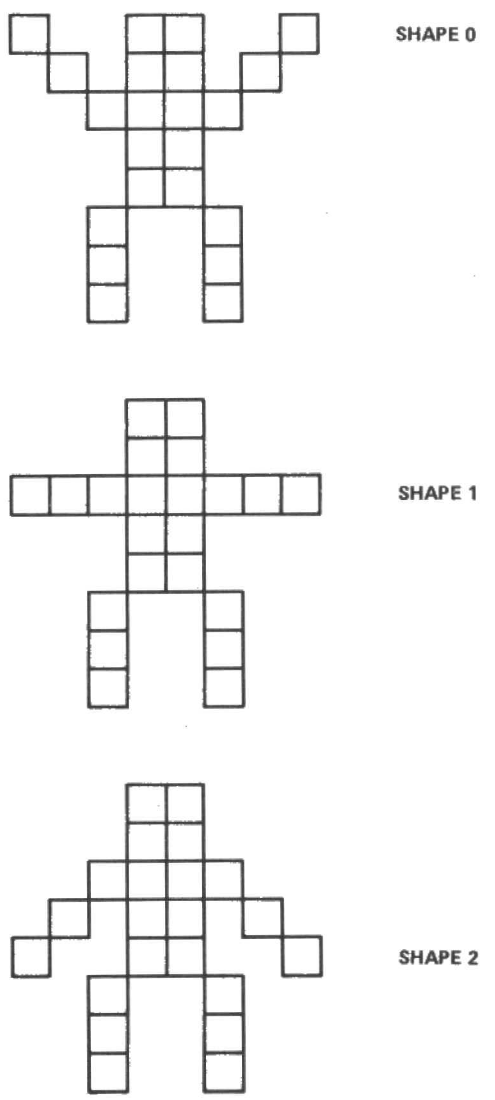


Figure 3. Internal animation of a sprite using a sequence of three shapes.

USING A LOGIC PULSER

Harry Fairhead has built up GSC's Logic Pulser kit and describes how to make the best use of the instrument.

Finding faults in digital circuitry is a task that is made a lot easier by having the right equipment. Fortunately the basic equipment needed for testing digital circuits is a lot cheaper than the main piece of test equipment needed for testing analog circuits – an oscilloscope. You can quite happily find faults and generally debug digital hardware with only two instruments – a traditional multi-meter for checking power supplies and any transistors that might have found their way into the circuit and a logic probe for checking that the digital part of the circuit is working.

Although in principle a logic probe can be used to find most faults in a digital circuit it can take both time and careful thinking to pin down the cause of an incorrect logic state. The trouble is that a logic probe can only tell you the logic levels at the various points of a circuit – it can't tell you what is causing a given state. For example, if you observe that an output to a gate is high when it should be low then obviously the suspicion falls on the gate that is driving it. The fault might be due to the output of the driving gate being stuck high or to the input of the driven gate holding the output high. The fault could lie in either of the two gates or, in a complicated circuit, it could lie in a gate further away in the circuit. The easiest way to find out if a gate is working is to monitor its output while a pulse or a stream of pulses are applied to its input and this is where a logic pulser comes in handy. A logic pulser will supply either a single pulse or a stream of pulses that can be injected into any point of a circuit. In this sense a logic pulser is the natural companion of a logic probe – the pulser injects pulses and the logic probe looks for them!

There is no question that a digital pulser is a useful addition to any collection of test equipment but unless you are going to be involved in a lot of fault finding the extra expense may be difficult to justify. All the more reason, therefore, to consider a digital pulser in kit form. Building test equipment from kits is rewarding both in the economic sense and in the pleasure to be gained from putting something together.

The GSC Kit

GSC offer a range of ready made logic probes and kits all using the same convenient plastic case. The GSC logic pulser kit is no exception and once complete the only problem is to make sure that you pick up the correct grey plastic case from the bench! The kit comes packaged in one piece on stiff card. Inside, along with all of the components, the case and all the hardware required, is a 12 page instruction manual. This gives not only very clear constructional details but a good account of how the pulser works and a full circuit diagram.

The actual process of assembly is quite easy and can be accomplished in about 1 to 2 hours – provided that you have taken the trouble to read the instructions beforehand. All of the components are easy to identify and the single sided printed circuit board presents no special soldering problems. However the components are mounted quite close together to allow everything to fit into the slim case and so you have to exercise a reasonable amount of care. The only really tricky bit is that the printed circuit board isn't silk-screen printed so you have to make reference to two component placement diagrams to find out where everything goes. While this is not too difficult it is worth double checking the position of each component before soldering it in place.

As long as you do manage to get everything correctly positioned the logic pulser should work first time as there are no critical components and no adjustments to make. You can check that the pulser is working by examining its output using its matching logic probe (after all there is no point in having a pulser unless you have a logic probe!) Just in case there is any trouble the construction manual also gives some helpful hints on why the pulser might not work. However as the logic pulser that I put together worked as soon as I applied the power I cannot comment on exactly how useful they are!

In conclusion, the GSC logic pulser kit is very professionally produced and as well as being an enjoyable way of adding to your test equipment it is also a way of saving money without sacrificing quality.

In Use

When confronted by a faulty circuit board the first thing to do is to check that the supply voltages are correct. This serves both to eliminate one of the most common reasons for circuit failure and to identify suitable points to attach the power leads of the logic probe and pulser. This job can often be more difficult than it sounds because not many manufacturers think of providing convenient places for even miniature crocodile clips! Once the power connections have been seen to, testing can then begin.

The first stage of any fault finding involves the use of the logic probe to check the logic levels at points of the circuit where the fault is thought to lie. In some cases a damaged chip can be located immediately because it has an output pin that is at an intermediate state – ie neither high nor low. Apart from one or two very special cases the output of a working logic gate has to be either high or low, so an intermediate state is a good indication that something is wrong.

If all of the outputs are at one of the two logic levels then there is no choice but to analyse the circuit more carefully. For example, suppose that the simple circuit shown in **Fig 1** was under test and the logic probe revealed that each point of the circuit were at the logic states also shown on the diagram. The first nand gate marked A is clearly not working properly because two high inputs to a nand gate should result in a low output not a high output as shown. The fact that gate A isn't working is easy to see with the logic probe alone but what about gate B? This is clearly working as the output is correct for the

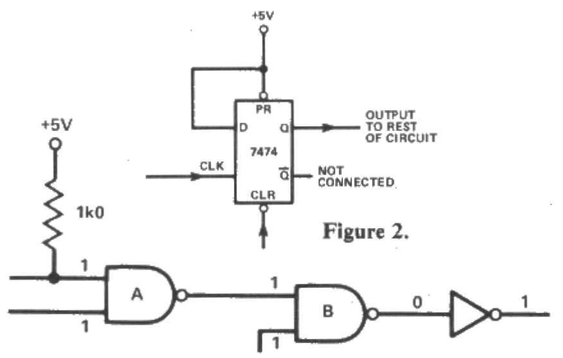


Figure 2.

Figure 1.

inputs shown. However this doesn't mean that the gate is working – it only means that you have no evidence that the gate is faulty. It is important to realise that, while finding an incorrect logic level is a sure sign that something is wrong finding a correct logic level doesn't mean that everything is fine. For example, it might be that the output of the nand gate marked B is simply stuck low and so will remain low even if the inputs change! Any output or input that you have only seen at one logic level is still a suspect in your search for faults.

A positive proof that gate B is working could be obtained by applying a negative going pulse to either of its inputs and checking that the output changes. This is of course where the logic pulser comes into its own. By applying the tip of the logic pulser to one of gate B's inputs and pressing the button on the pulser, the input can be made to change state momentarily. If while all this is going on you can manage to hold a logic probe on the gates output you should see it change state. If you don't then it is most probably faulty. It must be admitted that working on a circuit with both logic pulser and logic probe does need a certain amount of manual dexterity – eating chinese food with chopsticks is good practice!

How It Works

You may be a little worried about the exact way that a logic pulser injects pulses. After all, if you place the pulser on the output of gate B and press the button it will inject a pulse into the input of the invertor. This will pull the input high and, if the invertor is working, you will see a pulse at its output. This isn't at all worrying until you think about what happens to the output of gate B because, during the pulse that drive the invertor's input high, the output of gate B is also driven high against its will. Normally, if you force an output to a voltage level that it doesn't want to be at the result is smoking silicon! The reason for this is simply that the output stage draws or sinks so much current in an effort to maintain its proper output level that it cannot sustain the power output for any length of time without overheating. The key to the way that the logic pulser works without damaging the output of any chips is that it applies a pulse for such a short time interval that the output stage doesn't have time to overheat.

There is one way in which a logic pulser can fail to work. If the output stage that is driving the input that you are testing has a very low impedance then it is possible that the logic pulser will not be able to change its state. The most obvious example of this is where an input has been tied high or low by direct connection to one of the supply lines. In this case the only way that a logic pulser could alter the inputs state is by destroying the power supply! A less extreme case occurs when an input is tied high through a resistor of around 1 k. Once again in this case the logic pulser may not be able to sink enough current to bring the voltage level down. In practice this inability to change the state of an input is easy to detect using the logic probe. For example, suppose the tip of the logic pulser was placed on the input of gate A that is connected to the 1 k resistor. You may fail to detect a change in the output of the gate as a result of injected pulses. However before concluding that the gate was at fault it is advisable to place the logic probe onto the same input as the logic pulser to make sure that its state really is changing. Of course if you discover that the pulser cannot change the state of an input then this itself is useful information in your fault finding.

Sequential And Complex Circuits

Generally speaking finding faults in circuits that involve simple gates as showing in **Fig 1** is fairly simple and just a matter of checking logic levels and injecting pulses to make sure that gates are working. Unfortunately most digital electronics is not static but sequential. That is, in most cases, the circuit is continually changing its state in response to a stream of clock or external driving pulses. Testing a sequential circuit is mainly a matter of checking with a logic probe that the clock pulses pass through the circuit and do cause the changes that you would expect – ie shift registers shift, counters count and flip flops both flip and flop. The trouble is that sometimes a clocked sequential circuit comes to a complete stop and without any clock pulses it is difficult to check which component has failed and hence which component is killing the clock pulses.

Once again the solution is to use the logic pulser. This time to inject a stream of pulses to try to activate the circuit. This can sometimes be a little difficult if more than one source of clock pulses is missing but often injecting a stream of pulses to replace one clock can make the circuit come back to life and so tell you which clock is most involved in the fault. It is difficult to be specific about how to use a pulser and logic probe combination to test sequential circuits because the range and subtlety of the types of fault that can be encountered. However, a typical problem can be seen in **Fig 2.** This is a real example taken from a printer circuit. It shows a D type flip flop with its D input tied high. Its normal mode of operation is that the Q output remains high unless a pulse on the CLR input sets it low where it remains until the next clock pulse sets it high again. When the entire circuit, of which this D type flip flop is a part, is working correctly, CLR pulses happen quite often and as a result the Q output is pulsing continuously. The state that it was found in after the equipment failed was that the CLK pulses were present but the CLR pulses, and hence the Q output pulses, were missing. The obvious conclusion to come to is that some other part of the circuit is faulty and responsible for cutting off the supply of CLR pulses. This is a very reasonable assumption and if you were only using a logic probe it would take some time investigating other, and possibly complex, parts of the circuit before coming back to re-examine this simple D type flip flop. With a logic pulser testing the flip flop and ruling it out of consideration as the source of the fault is very easy. By injecting a stream of pulses into the CLR input and monitoring the Q output for pulses is a sure check that the D flip flop is working.

In the circuit in question it was discovered that the flip flop wasn't working and the reason that the CLR pulses had vanished was that they were in fact nothing more than a gated and shifted version of the Q output itself! In other words the D flip flop was the faulty part of a feedback circuit. A further proof that this was the case was obtained by injecting a pulse stream into the Q output which caused the CLR pulse to reappear.

This example illustrates how a logic probe and pulser used together can provide positive proof that a component is or isn't working. Without this sort of information there is no alternative but to use the logic probe and logic to work out which component is at fault, according to the state of the circuit.

The Benefits

Since I added a logic pulser to my collection of test equipment, which alos includes more sophisticated items such as a scope and a pulse generator, I have been surprised by the number of times it has proved useful. It is more convenient and easier to handle than the more sophisticated test gear and it enables faults to be found very rapidly. By making the collection of positive evidence that a chip has failed easy to collect, it has managed to cut down on the number of times that I have needlessly replaced a chip – and anything that cuts down the amount of surgery that a PCB is subjected to is a good thing!

I have also found that some unusual applications for the logic pulser. For example, the other day a computer system suddenly stopped printing – did the fault lie in the printer or in the computer itself? The answer was quickly found by injecting a stream of pulses into the STROBE line of the centronics interface. As a result the printer burst into life printing garbage but at least it showed that it could print. Five minutes more work with the logic probe and the pulser identified the fault as a damaged buffer chip.

As you can guess I am very pleased to have at last made the acquaintance of a logic pulser – my only regret is for the wasted hours I must have spent looking for faults without it.

E&CM

CIRCUIT ANALYSIS PROGRAM

Mike Rigby assesses Number One Systems' circuit modelling package for the BBC microcomputer.

Circuit analysis by computer is a facility which, until recently was available to relatively few people, and at very high cost. With the rapidly decreasing cost of small microcomputers, and the rapid spread in their usage, this situation is fast changing. The modern microcomputer is ideally suited to this type of application. By using matrix nodal analysis techniques, a system can be evolved for circuit analysis which is ideal for use on a computer, allowing the rapid analysis of quite complex circuits.

This article describes a circuit modelling program (referred to from now on as Analyser) which can be of great help to those involved in the design of linear AC circuits. Possibly the most obvious application of a program such as this is that of filter design, although filter circuits are by no means the only ones which lend themselves to linear analysis. Filter circuits of moderate complexity can be modelled within virtually any frequency range from near DC to almost infra-red.

The one major advantage of computer modelling is speed. To model the example filter circuit shown later in the article would take approximately ten minutes between running the program and examining the results. To build the circuit (assuming everything was to hand, including all the correct components) and then measure its performance (collecting the equipment together first) would take far longer. A further advantage of a program such as this lies in its precision. If one plots the effect of varying one component value, that effect can be seen very precisely, allowing the mathematical equations which model the behaviour to be seen far more easily than is the case if one draws conclusions just from measurements. A lecturer in an educational establishment with access to perhaps less than perfect test equipment could use the program in preference to a practical demonstration to illustrate a lecture.

Linear Techniques

Analyser is a numerical modelling program which allows the Gain, Input Impedance and Output Impedance of a linear electronic circuit to be investigated. The program generates its output in tabular form, displaying the magnitude and phase of the parameter selected for analysis.

The program deals with linear components, and can therefore deal only with those components which can be represented by a linear model. Models of resistors, capacitors, inductors, transformers, bipolar transistors, field effect transistors and operational amplifiers are provided.

Bipolar transistors and field effect transistors are non-linear devices, and have a performance which cannot be modelled with complete accuracy. However, if the assumption can be made that the signal is small enough to cause no non-linearity, then the device can be modelled accurately enough to predict circuit behaviour. Although operational amplifiers are not linear devices, in practice, they exhibit highly linear behaviour when used with the high levels of negative feedback which are normally employed.

Analyser requires a Model B BBC microcomputer or a Model A with 32K of RAM. The program is fully compatible with both OS 0.1 and OS 1.2. Although the program is supplied on cassette, it is fully compatible with discs. When Analyser is run, it checks which operating system is fitted. If OS 0.1 is fitted, the cassette operating system "bugs fix" program is assembled into memory to allow 1200 baud Data files to be used. If OS 1.2 is in use, the program checks whether tape or disc storage is currently in use and alters it's mode of operation accordingly.

The program is fully menu driven, and is therefore broadly self-documenting. When in main menu mode, operating system commands such as *DISC, *TAPE, *OPT etc. may be used, allowing for example the program to be loaded from disc, but ensuring that previously saved data files can be read from tape. It is also possible for example to configure the operating system for use with serial printers (using *OPT).

In Operation

As a general principle when using the program, nothing need by typed unnecessarily – if the program already has a piece of information (eg the start frequency of the analysis), then when the user is prompted for this piece of data, he/she need type only Return to re-enter the previous value.

The program can handle circuits of up to 16 nodes and 60 components (where a node is the junction of two or more components).

When the program is run, the user is prompted for the date. This will be used when the circuit values are stored as a data file on tape or disc.

Following date entry, the user will be asked whether or not they wish to use a DATA statement (held at the end of the program) to enter the component values. This facility will not normally be required, being included principally to allow convenient demonstration of program operation. If the user chooses not to use the DATA statement to enter values, the main menu will be presented and the user will be given a list of options **(Table 1).**

The normal choice if a new circuit is being designed is <1> Start new circuit. The program will prompt the user for a circuit name (which will be used when saving the circuit data on Tape/Disc). Following name entry, the components may be entered. Component values are entered in a systematic nodal notation, with the program checking for erroneous entries (eg negative node numbers or component values). Capacitors and inductors are entered exactly, but resistors each have 0.2 picofarads of capacitance added across them. This value is representative for a 0.25 Watt carbon film resistor. The component values are entered as follows:-

Resistors, capacitors and inductors:-
These are entered by specifying a component name (R1, R2, C1, C5, L4) followed by the nodes between which the component is connected, followed by it's value.

Field effect transistors:-
FETs are entered by specifying the Gate node, the Drain node, the Source node and the transconductance (gm) in Amps/Volt.

Bipolar transistors:-
Bipolar transistors are entered by specifying the Base node, the collector node, the Emitter node, the h_{fe} and the Collector current in mA.

Operational amplifiers:-
Op-amps are entered by specifying the positive input node, the negative input node and the output node.

Transformers:-
Transformers are entered by specifying the four nodes between which they are connected (two on the primary side, two on the secondary), followed by the turns ratio. Winding resistance is dependent on the application and should be added to the circuit diagram as a discreet resistor.

Once the components have been entered, the entry procedure is terminated by component 'P'. P is the end of file marker, and is used to enter the input, output and ground nodes. Unlike other such programs, a totally free choice of node numbers is allowed for input, output and ground. The numbers selected may be altered by using the editing procedure, in the same way as altering any other component.

Example Session

The values are entered in response to screen prompts: eg for resistor R1 between nodes 0 and 1 and of value 10k ohms, and resistor R2 between 1 and 2; value 100 ohms, entry would be as **Table 2.** ▶

When component entry is complete (following entry of 'P' and the input, output and ground nodes), the component list is displayed for inspection and a return is made to the main menu to allow component editing, or an analysis to be run etc.

As an example, consider the circuit shown in **Fig 1.** This circuit is that of a 12dB/octave high pass filter and would be expected to have a frequency response which rises at 12dB/octave at low frequencies, levels out at unity gain, then drops in gain at high frequencies as the open loop gain of the operational amplifier falls off.

The component entry procedure for this circuit would be as shown in **Table 3.**

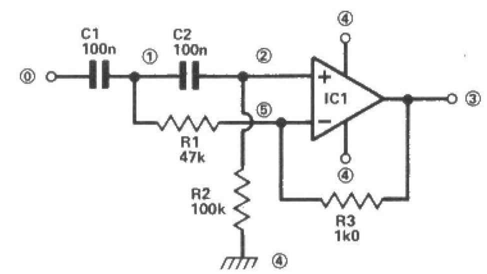


Figure 1. A 12 dB/octave high pass filter that lends itself to analysis by the circuit modeller program. Table 3 shows the entry of component values while the example program outputs show the circuits behaviour.

When the list has been checked, a return can be made to the main menu to allow the next action to be selected. If circuit modification is necessary, option <2> should be used. The component entry procedure will be invoked once again, and when all changes are complete, the main menu will once again appear. If an analysis of the circuit is required, main menu option <3> should be chosen whereupon the screen will clear and the desired frequency range can be entered in response to the screen prompts as follows:-

Number of steps (– for log (50max), D for Data)?

If a positive number N (less than 51) is entered, the required frequency range will be split into N equal steps when the analysis is run. If a negative number is entered, the frequency range will be split logarithmically. Since the analysis frequencies are calculated, they tend to give frequencies such as 837.28Hz. This is inconvenient for graph plotting, so a further option is offered: that of a Data statement which covers the audio range in twenty convenient steps. If Return is pressed in answer to the prompt, the previously used value is re-used with the default value equal to –10 (10 logarithmic steps).

When the number of steps has been entered, the program will prompt for a Start frequency (default is 1 Hertz), followed by an End frequency (default is 1 Megahertz). Analyser will now offer a menu as shown in **Table 4.**

If Input or Output impedance are being plotted, the word GAIN is replaced by Zin or Zout respectively. If <2> is chosen, the program requests a nominal Gain/Zin/Zout in dB. If <4> is chosen, a nominal value in Volts/Volt (Gain) or Ohms (Zin/Zout) is requested. The analysis now starts, the results appearing on the screen as they are calculated. Once the results table is complete, the user is offered the option of printing the component list and results on a printer, then another run is offered (allowing a different frequency range or number of steps). If the reply to the last prompt is 'N', the program returns to the main menu to allow component modification or whatever.

If main menu option <4> (Change analysis parameter) is selected, a menu is offered allowing a choice of options as in **Table 5.**

When a choice has been made, the main menu returns. It should be noted that pressing ESCAPE will at all times cause a return to the main menu.

Conclusion

This article has set out to demonstrate that, although not all design problems can be solved by using nodal analysis, there are nonetheless many situations where this program could prove valuable. It would probably be true to say that if you are involved in AC circuit design, and you do not have access to this facility, you are placing yourself at a disadvantage as compared with those people who have.

E&CM

<1> Start new circuit
<2> Modify circuit
<3> Analyse circuit
<4> Change analysis parameter
<5> Load circuit off Tape/Disc
<6> Save circuit on Tape/Disc
<7> Change circuit name
<8> Catalogue Tape/Disc

Table 1. The various options open to the user of Analyser are shown in the program's Main Menu.

```
COMPONENT NAME?R1
?0
?1
?10000
COMPONENT NAME?R2
?1
?2
?100
```

Table 2. Component values are entered in response to a series of prompts.

```
COMPONENT NAME?R1
?1
?5
?47E3
COMPONENT NAME?R2
?2
?4
?1E5
COMPONENT NAME?R3
?3
?5
?1000
COMPONENT NAME?C1
?0
?1
?0.1E-6

COMPONENT NAME?C2
?1
?2
?0.1E-6
COMPONENT NAME?A1
?2
?5
?3
COMPONENT NAME?P
?0
?3
?4
```

Table 3. The component entry procedure for the circuit shown in Fig 1.

<1> GAIN (dB ABSOLUTE)
<2> GAIN (dB RELATIVE)
<3> GAIN (LINEAR ABSOLUTE)
<4> GAIN LINEAR % ERROR)

Table 4. The menu options displayed after component values and frequency range have been entered.

<1> VOLTAGE GAIN
<2> INPUT IMPEDANCE
<3> OUTPUT IMPEDANCE

Table 5. The choice of options available if main menu option <4> (change analysis parameters) has been selected.

The circuit analysis program is available from Number One Systems at 9A Crown St., St. Ives, Huntingdon, Cambs at £35 for the cassette version.

The company also produce a version for the 48K Spectrum.

CIRCUIT NAME **Filter** 17.8.83.

Component list:

R1	1	5	47000
R2	2	4	100000
R3	3	5	1000
C1	0	1	1E-7
C2	1	2	1E-7
A1	2	5	3
P	0	3	4

TEST RESULTS

Frequency		GAIN(dB abs)	Phase
5.00	Hz	-26.637	166.67
10.00	Hz	-14.678	145.99
20.00	Hz	-4.237	103.28
40.00	Hz	-0.306	50.68
80.00	Hz	0.013	23.72
160.00	Hz	0.009	11.61
320.00	Hz	0.002	5.76
640.00	Hz	0.000	2.87
1.28	kHz	-0.000	1.42
2.56	kHz	-0.000	0.67
5.12	kHz	-0.000	0.26
10.24	kHz	-0.001	-0.02
20.48	kHz	-0.001	-0.31
40.96	kHz	-0.001	-0.75
81.92	kHz	-0.004	-1.58
163.84	kHz	-0.013	-3.18
327.68	kHz	-0.050	-6.36
655.36	kHz	-0.196	-12.60
1.31	MHz	-0.736	-24.22
2.62	MHz	-2.413	-42.71

Sample program output showing gain vs frequency for the Fig. 1 circuit.

CIRCUIT NAME **Filter** 17.8.83.

Component list:

R1	1	5	47000
R2	2	4	100000
R3	3	5	1000
C1	0	1	1E-7
C2	1	2	1E-7
A1	2	5	3
P	0	3	4

TEST RESULTS

Frequency		Zin(abs)		Phase
5.00	Hz	314.358	k	-81.19
10.00	Hz	153.564	k	-70.50
20.00	Hz	82.844	k	-42.88
40.00	Hz	79.066	k	-9.95
80.00	Hz	91.958	k	-0.55
160.00	Hz	97.738	k	0.44
320.00	Hz	99.422	k	0.29
640.00	Hz	99.860	k	0.10
1.28	kHz	99.970	k	-0.06
2.56	kHz	99.997	k	-0.26
5.12	kHz	100.000	k	-0.59
10.24	kHz	99.987	k	-1.22
20.48	kHz	99.925	k	-2.45
40.96	kHz	99.679	k	-4.90
81.92	kHz	98.711	k	-9.74
163.84	kHz	95.095	k	-19.04
327.68	kHz	83.701	k	-35.16
655.36	kHz	60.130	k	-56.81
1.31	MHz	33.660	k	-76.61
2.62	MHz	15.878	k	-88.22

Tabular output showing input impedance vs frequency for the 12 dB/octave high pass filter.

OPTIMAL CODING

Part 2

Last month James Dick looked at a number of data compression techniques. This month he looks at pre-coding and describes some applications of the principles of optimal coding.

The codes mentioned last month provide a means of reducing the number of characters required to convey a set of messages. It should be realised however that a certain amount of pre-coding also helps in this task.

Altered Images

When processing images for transmission, Run Length Coding (RLC) is a technique that is often used. When an image is scanned, it consists of a series of horizontal scans in each picture frame – rather like a conventional television picture. If the intensity is plotted against horizontal distance for a particular scan, it is often found that a series, or 'run', of picture samples, 'pixels', has the same intensity. **Fig 1** details this. Instead of transmitting the picture as an m x n array of intensities, it may be transmitted as RLC pairs. The first word representing the position in a scan of a run start, and the second word representing the intensity level of that run.

Another technique, perhaps more general, is that of Differential Pulse Code Modulation (DPCM). Here, the difference between the i-th and the (i + 1)th sample in a data stream is coded rather than the absolute value of the (i + 1)th sample. In most cases, signals change rather slowly compared to the sampling interval so DPCM reduces the size of the data-steps to be coded. **Fig 2** shows a typical waveform while **Fig 3** details the hardware configuration for the encode and decode processes. The 'quantiser' has the task of digitising the difference between the actual input and the predicted input. When the input signal is changing rapidly and the jump between samples exceeds the range of the quantiser, 'slope overload' is said to occur while the inability of the quantiser to follow small variations leads to 'granular noise'.

When the quantiser uses only one bit, the DPCM process is called 'delta modulation' and the fidelity of the recovered data tends to be poor compared with quantisers using several bits.

In all cases, optimal coding has to be undertaken with care if true minimising is to be achieved. It may be necessary for several different techniques to be tried before the ideal one is found. In a long data stream, adaptive coding may be used. This involves splitting the data into subgroups and assigning a coding technique to each subgroup.

Applications

Returning to the topic of language raised at the start of last month's article, text is particularly suitable for optimal coding. It is nearly impossible to estimate the number of messages that are transmitted in machine coded form (eg RTTY) every day. Most RTTY traffic will use 26 alphabetic characters, 10 numbers and an assortment of others (eg SPACE, ?, /, WHO-ARE-YOU,....). If English alphabetic text *en clair* is statistically analysed, certain characters are much more common than others. **Fig 4** shows the distribution for some 25000 characters. A quick estimate implies that 26 alphabetic characters (and a space) need 5 bits to transmit. Indeed, the international 5 unit Teleprinter Code, **Fig 5**, is such an example. Using a minimum length code, the alphabetic characters may be equivalent to the code words shown in **Fig 6.** The efficiency of this system is 99.4% the average

RUN LENGTH CODING

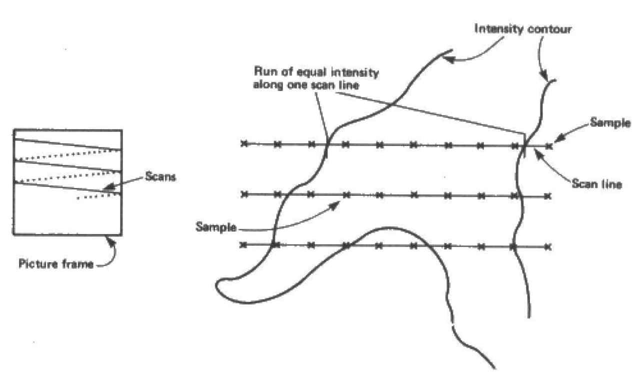


Figure 1. RLC techniques are often used when transmitting images. Instead of transmitting an image as an array of intensities it is transmitted as RLC pairs, the first word determining the start of tile run and the second the image intensity.

DIFFERENTIAL PULSE CODE MODULATION

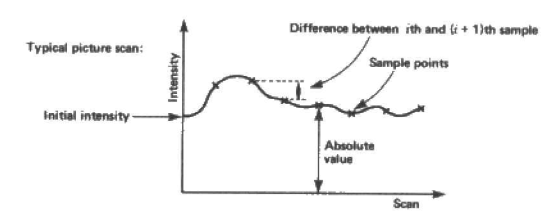


Figure 2. DPCM systems transmit an analogue signal, not as a series of words representing the absolute value of the signal at any point in time but as a sequence of difference values between the i-th and (i + 1)th samples.

DPCM SYSTEM

Figure 3. A much simplified block diagram of the hardware required to implement a DPCM system.

STATISTICAL DISTRIBUTION OF TEXT

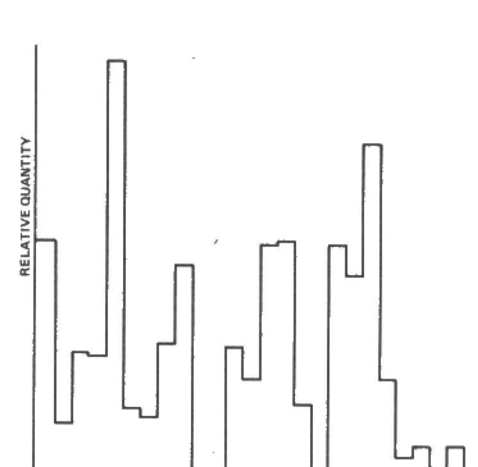


Figure 4. A histogram showing the statistical distribution of the 26 alphabetic characters based on a sample of RTTY traffic. It can be seen that certain characters (eg E, T and A) are much more common than others.

word length is just over 4 bits, implying a saving of nearly 20% compared to the straight 'binary' type code. It is interesting to compare the transmission time of Morse code with the minimum length code. Note that the codes discussed here differ from Morse; the latter uses a stop-bit of three units to indicate the end of a code word.

Practical Applications

Remotely Piloted Drones (RPD) are often used as aerial targets or for reconnaisance. They are small in size and cost; the former ensures suvivability because of the small cross section presented to missiles while the latter quality allows many to be built. RPDs send back a stream of data as well as a TV image of where they are – this is the feedback to the pilot. While normal quality TV produces data at around 15 Mbytes per second drones use pre-coding to reduce the number of frames sent per second to (perhaps) one and to reduce the spatial and intensity resolution. Optimal coding is then used to further reduce the total quantity of data transmitted. Data compression by factors greater than 30-to-1 may be achieved.

Satellites tend to produce large ground-based data storage problems. An Earth resource satellite taking many multi-spectral images per day may amass more than 200 Gigabits per year. Optimal coding allows more data per warehouse without losing any information.

Speech synthesiser systems often use code compacting to allow efficient storage of the data detailing the complex waveforms used. The compacted data also leads to fast searching when a computer is listening to speech and trying to recognise words.

In Conclusion

Last month we made the point that, with an increasing need to transfer ever increasing amounts of data via communications links that offer only limited bandwidths, the techniques described in this article will become more important over the next few years. We have shown that, with fairly straightforward analysis of the data to be transmitted and applying a suitable coding system, dramatic improvements in the efficiency of a communications link can be readily realised.

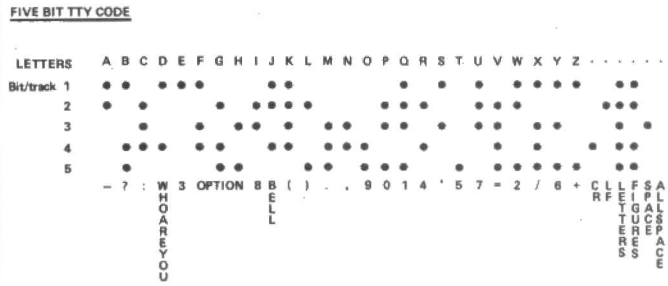


Figure 5. The international teleprinter code is a 5 bit code that allows the 26 alphabetic characters, a space and various control functions to be encoded. This form of code is not the most efficient and the techniques described in the text can lead to savings of up to 20%.

HUFFMAN CODE FOR ALPHABET FROM TEXT SAMPLE

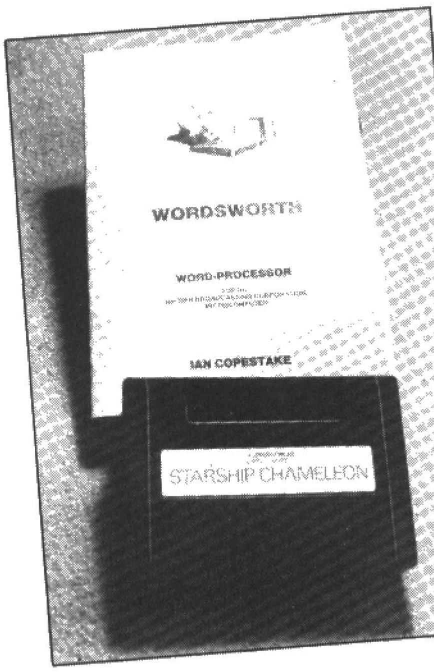
AWL = 4.20
Entropy = 4.18

A	0001	N	0100
B	001111	O	0010
C	00001	P	10111
D	00110	Q	10110011
E	100	R	0101
F	11100	S	1010
G	001110	T	110
H	1111	U	01110
I	0110	V	1011000
J	101100100	W	111010
K	1110111	X	1110110
L	00000	Y	101101
M	01111	Z	101100101

Figure 6. The code shown here has an average word length of just over 4 bits and leads to a far more efficient transmission system when compared to one adopting a straight 'binary' type code.

The vast array of books and software currently available presents the microcomputer user with a bewildering choice. This page is intended to help readers by presenting an informed and considered opinion about a couple of books and pieces of software each month. Because space is limited, this review page will be selective, and only books that seem to offer a good deal will be included. Every book reviewed on this page can be obtained through the ECM Book Service.

Each month Harry Fairhead will look at a selection of recently published books and software.



Wordsworth
for the BBC Micro Model B
Ian Copestake,
23 Connaught Crescent, Brookwood, Woking, Surrey, GU24 0AN

This piece of software was selected on the grounds that it was inexpensive. This is obviously a point of recommendation of anv package but is not the only consideration when choosing something as potentially useful as a word processor. Wordsworth comes with clear and straightforward documentation in the form of a 22 page booklet and a prompt strip to insert under the clear plastic strip at the top of the BBC Micro's keyboard. This is to remind you of the role of each of the red function keys which are used by the program. The package offered most of the facilities required, both the ones that have come to be seen as standard – justification, line centering and finding and replacing specified strings – and others that might be seen as extras – a word count option, for example. There were however respects in which it was difficult to use to the point that you felt an option was worthless. This was particularly true of the change case option which was so confusing that it was preferable just to overwrite the letters to be altered. The worst feature, however, was the lack of an easy way of inserting anything into existing text. To insert even a single character the AMEND option was required which was accessed by pressing CTRL A at the exact point where the amendment was to start. This had the effect of making ALL text from that position to the end of the file disappear. You then have to rely on your memory while making the insertion – unless of course you'd already printed out the text. Once the insertion is completed pressing CTRL A restores all the text at the current cursor position reformatting it as it does so. If your file is long it can be quite time consuming to make even minor changes. The manual suggests moving any passages with "awkward amendments" to near the end of your test, making changes and then moving it back. My experience is that I make little omissions of letters and words throughout a text and want to make lots of changes when I re-read so I soon got frustrated by this procedure which left me in the dark!

In conclusion, Wordsworth turned out to be false economy and that if you have any long lasting use for a word processing package this one just isn't powerful enough. It is written mainly in BASIC and this makes it slow and cumbersome. These comments apply to the cassette version and while the disk version does offer some additional features – they do not overcome its fundamental limitations.

Starship Chameleon
Dragon Data Limited

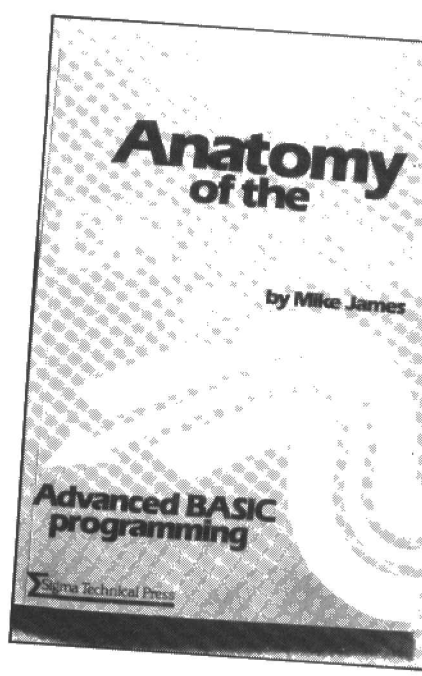
For a bit of light relief this month, we look at one of the favourite games for the Dragon.

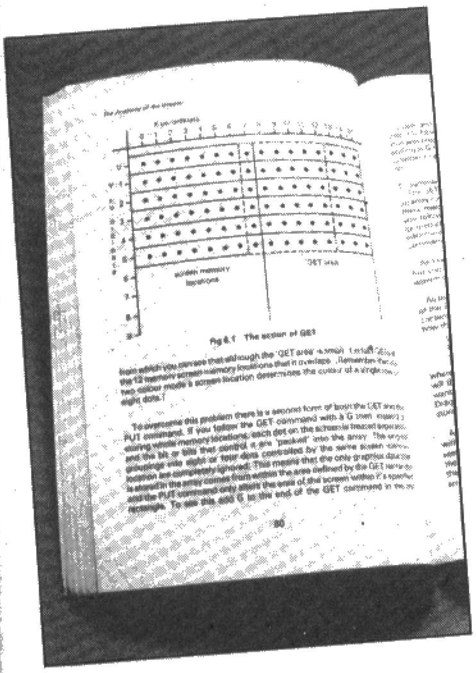
The game's ingredients are fairly familiar ones – bombs falling out of enemy craft which have to be intercepted and other missiles which are absolutely deadly. There is just a bit more to it than that, however. The player's "Starship Chameleon" can change colour from yellow to blue and vice versa. While yellow it can destroy yellow bombs but will be destroyed by blue ones and while blue it can destroy blue bombs but will be anihilated by yellow ones. The extra hazard is the aerial mines which "are programmed to pursue and destroy your ship" and which cannot be destroyed, only avoided. The objectives of the game are firstly to survive – you have three ships and an extra one for every 10,000 points scored – and secondly to maximise your score. There are two classes of bombs – bombs which count 200 points when you hit them and – 100 when they hit you and superbomb which scores 1000 plus or minus.

The game is played with joysticks. This leads to a slight time lag between the player's response and what actually happens but it's quite easy to get accustomed to this. The game can be played by one or two players. Once you've selected one or two players you choose the difficulty level, between 1 easy and 4 difficult. If two people play they take alternate ships and the competing scores are displayed on the top line. There is a high score table displayed at the end of each game. One unusual feature of this game is that you can freeze if in mid-game by pressing the P button. A useful feature for those players who get so good at the game that they carry on for hours.

Advanced Graphics with the Sinclair ZX Spectrum
by I. O. Angell and B. J. Jones
Macmillan, 1983

This book represents a great deal of hard work on the part of the authors. It contains a lot of programs which are built up from modules. This may be rather confusing until you get used to the idea that to replicate the illustrations shown you need to collect together routines given at various points in the book. Naturally, the programs that produce sophisticated pictures are lengthy but there is a short cut to all the typing as there is a cassette tape available. Most of the techniques covered are for two-dimensional and three-dimensional line drawings. Lots of the two-dimensional results are decorative and many have useful applications – histograms, pie charts, diagrams and graphs. Three-dimensional representation on a two-dimensional screen is necessarily a sophisticated topic and it is easy to become overwhelmed by the time this part of the book is reached. Even so it was pleasing to find a program for a general hidden line algorithm – something that is definitely difficult to produce from scratch. Most of the book is





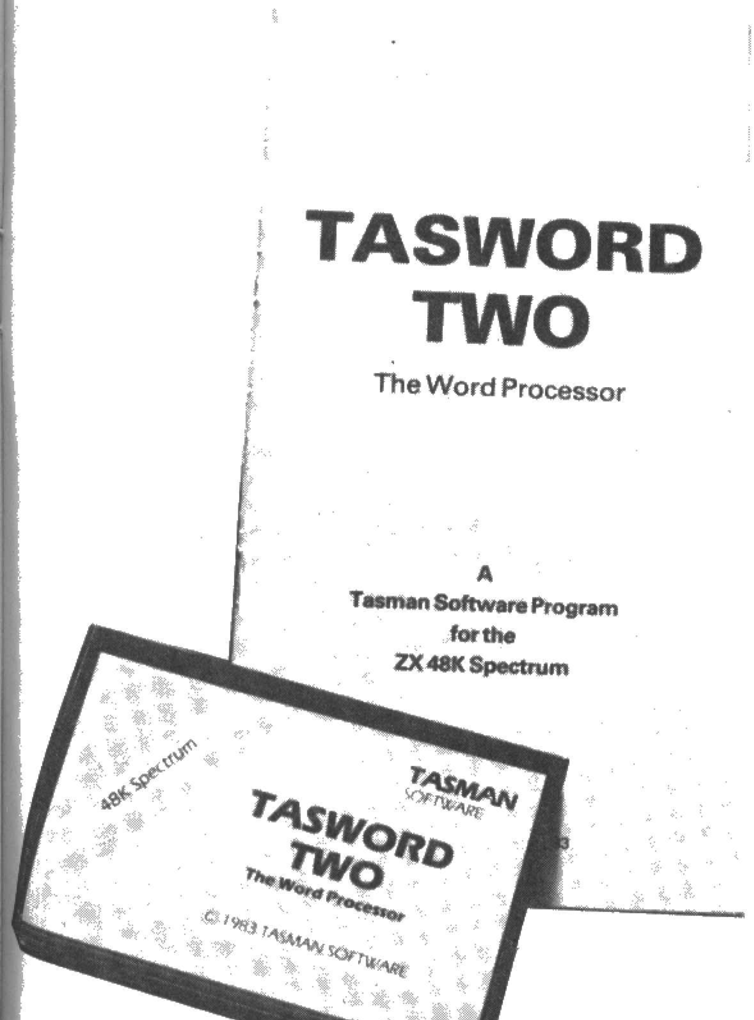
devoted to static graphics but the last but one chapter presents an animated graphics game – for the 48K Spectrum only. Compared to the level of detail given in relation to other techniques the explanation presented in this section seems rushed, and gave the impression that the authors were not as interested in this topic.

This book is an unusal mixture of the academic and the popular and is aimed at a diverse audience. Its authors claim that it can be used at different levels – as a source of graphics programs for fun or for more serious applications or, as they intended it, as "an introductory text to computer graphics" leading the reader from "elementary notions . . . to advanced topics". In this context, the word "introductory" is perhaps misleading. In fact to use the book as its authors intend you need to have a good grasp of cartesian geometry and matrix algebra. Without this understanding the chapters with titles like "Matrix Representations of Transformations on Three-Dimensional Space" are not likely to be at all enlightening. The book has been developed out of material contained in courses in computer graphics taught at undergraduate and postgraduate level and it doesn't pull any punches. If you can stay the course you will get a great deal out of it but if maths was never your strong subject then its value to you may be limited to using the programs. A BBC version has also been prepared by the same authors.

Anatomy of the Dragon
by Mike James
Sigma Technical Press, 1983

This book takes the reader on a voyage of discovery – an exploration of the Dragon microcomputer – with a real reward at the end of the journey – an understanding of how the Dragon works and how to get it to do what you want. My own brushes with the Dragon gave me a great deal of sympathy with what Mike James says in the preface to his book. He writes of his own early experiences, "things just didn't always work out as well as I expected and I just couldn't seem to produce the graphics effects that I wanted".

After a brief introductory chapter, the Dragon's anatomy – that is its block diagram and memory map – are introduced in Chapter 2. There then follow two chapters on graphics. The first of these reveals five extra "semi-graphics modes" not normally available but very handy once you know about them. To make sure you understand how to use them the chapter includes three subroutines and a program that calls them. The second adopts the same sort of approach to the "full graphics modes", again giving access to three that are normally hidden from the user. Chapter Five is devoted to sound and shows that the Dragon is a more versatile beast in this respect than is apparent from reading the manual. It is in the next chapter that this book really does to town on remedying the deficiences of the manual and explains paged graphics, the hidden depths of the DRAW command and the mysterious GET and PUT. The Dragon's interfaces are subjected to scrutiny in Chapter Seven and the structure of its BASIC is examined in Chapter Eight so and the book comes to a close with a brief look at assembly language. This serves as a taster for Mike James forthcoming book, "The Language of the Dragon" a companion volume devoted to this topic. If it's as informative a book as this one its publication (scheduled for October) is eagerly awaited.



TASWORD TWO

The Spectrum computer is hardly the first machine to spring to mind when it comes to word processing. S. M. Gee reports that with Tasword Two it is however possible to use a Spectrum as the heart of a very respectable WP system.

Tasword Two, at £13.90 inclusive of VAT, postage and packing, is a relatively inexpensive software package. It is supplied with a slim manual which looks as though it has been produced using the ZX Printer. This was one of the things that first aroused our interest as our own ZX Printer has been sitting idle for so long now that its print quality has degenerated which is a not uncommon fate for underused ZX Printers. The program itself takes less than three minutes to load and there is a second program on the tape, the Tasword Tutor, that was found to be extremely helpful. This Tutor introduces the various control keys that give access to Taswords facilities and persuades the user to try them out one at a time. After an hour of working our way through this demonstration we felt ready to use Tasword Two.

Getting started with Tasword Two is extremely simple. Initially there are little user-friendly touches that are reassuring to the novice user – messages telling you not to stop the tape which is flashed on the screen throughout the loading period and the one instructing you to stop the tape, accompanied by a beeping sound, once the load is completed. (The fact that the Spectrum lacks a cassette motor control means that it is more difficult to use for multi-loading operations than other micros).

Once the program has loaded you are faced with a screen which is blank apart from a flashing cursor at the top left and a status line at the bottom of the screen which gives vital information about the state of three options – justification, word-wrap and insert mode (of which more later) – and records the position of the cursor in the file. If it's the first time you have used Tasword you may feel intimidated by such an empty screen but if you follow the advice in the manual and load the tutor program you'll quickly gain confidence.

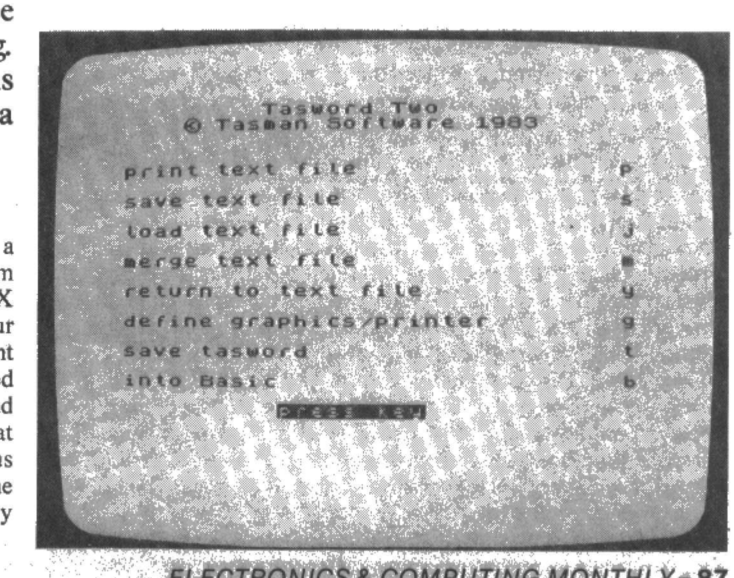
Entering Text

The first things the user wants from a word processor is the ability to enter and correct text easily. Entering text with Tasword Two is as simple as using the Spectrum's keyboard!! The keyboard's default mode is, as usual, lower case although upper case can be selected using the CAPS SHIFT or the CAPS LOCK in the normal way – when you select CAPS LOCK mode a message to the effect appears in the status line. Numbers are straightforwardly produced by pressing the number keys and punctuation marks and other single characters are also produced in the standard way – SYMBOL SHIFT has to be pressed for some like the ',' and '.' and '$', while EXTENDED MODE and SYMBOL SHIFT are required for some rather esoteric ones like square [] and curly () brackets and '@'. The delete key also works in the usual way but, as usual, we often managed to produce strings of 0s by pressing it on its own instead of with the CAPS SHIFT – which just goes to demonstrate what was meant by the cryptic comment about this software being just as easy as the keyboard. In practice, we found that entering text via the Spectrum's keyboard was somewhat easier than anticipated. The main difficulty was the slowness of the key response. Most of the keyboard problems could be solved by buying an add-on keyboard which would be well worth it for this application.

In Control

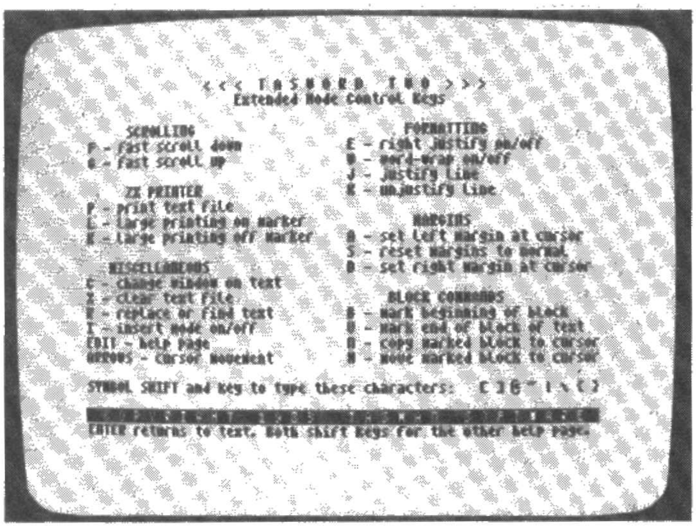
What makes a word processor more or less versatile it its control features. Tasword offers an impressive selection and the tutor program succeeds in demonstrating them admirably. All the combinations of keys on the Spectrum are brought into play one way or another. Some of the most immediately useful are the arrow keys – which move the cursor to any desired position – and the edit key which displays the 'help page'.

From the 'normal mode' help page it is possible to go either to the 'extended mode' help page or back to the text file. The two help pages contain instructions for all the control facilities and make reference to the manual unnecessary. To oversimplify matters a little, there are two distinct sets of control functions. One set uses the SYMBOL SHIFT markings – that is the words or characters in red on the letter



keys – using only the ones that have more than one character on them. This is very sensible as it in no way restricts the user's choice of characters to use in entering text. There are twelve such keys and between them they control moving to the top and bottom of the text, scrolling through the text a line at a time, centering text on a line or moving it to the left or right, deleting a whole line of text, inserting new material within existing text, reforming paragraphs after changes have been made, and getting out of text entry mode to print, save or load text files.

Let's deal with this last control option first. If you press STOP (SYMBOL SHIFT and A) you see the following menu:



All these facilities are very useful. The 'print' option allows you to send text via an interface to whatever printer you have. (If you are going to use a ZX Printer you don't need an interface, more about this later). When using a non-ZX Printer, you also have to go through another of the menu's options which is to do with printer initialisation. The 'define graphics/printer' option allows you to inform Tasword of the codes used by your printer for carriage return and line feed. Tasword Two comes already configured for the Epson FX-80 printer, which is very sensible since there are probably more of this model being used with Spectrums than any other non-ZX models, but the instructions for altering the codes (which rely on the Spectrum's graphics characters) to other values seem straightforward. Codes also need to be altered according to the interface being used. An insert to the manual gives instructions about using the following interfaces:

Cobra RS 232 I/O Port
Euroelectronics ZX Lprint
Kempston Interface
Hilderbay Interface

Tasman also market their own printer interface.

These parts of the program were not investigated but the instructions were clear. Text was printed to a ZX Printer – which uses a different command that is more immediate in that it doesn't go via this menu – when the program printed what we expected it to!!

Saving Grace

The 'save text file' option allows you to save a text file on cassette and the load option allows you to load such files. 'Merge' allows you to load a text file from tape and put it at the end of existing text. One note of warning here – this procedure will fail if the size of the old and the new text together exceeds 320 lines which is the maximum size for a Tasword Two file. There is no option for joining files together to create a longer piece of text.

The 'save Tasword' option allows you to save a back up or customised version of the program. The instructions for doing this are displayed on the screen. If you have redefined graphics symbols to suit your interface/printer combination than saving Tasword will ensure that you need not repeat this effort as all the codes you have set will automatically be set.

To return to editing text from this menu you select the 'return to text file' option. Alternatively, you can go straight into BASIC. I tried going into BASIC, performing a direct operation (i.e. without a line number) and returning to Tasword by typing RUN. This worked in that I found my text file just as I had left it. This might be useful if you wanted to do a calculation and put the result in the text you were preparing.

The next SYMBOL SHIFT control key worth discussion is the AND key. This enables you to insert between two lines, between two words, and between two characters. Insertion is a vital facility for any text processor and Tasword's method is fairly easy to use once you understand the limitation of 'Insert ON' displayed in the status line. All this means is that new lines are inserted automatically when you press ENTER. If you type in the middle of existing text without pressing AND you will overwrite, rather than insert text, at the cursor's position.

Once you have made alterations, your text will probably have rather a messy appearance. To rectify this, pressing STEP will cause the text to reform itself neatly. Position the cursor at the beginning of a line and all the text in the paragraph will be moved into an orderly position.

Although not all of the 'normal mode' control facilities have been described we're going to move on to the 'extended mode' control keys. As Spectrum users will be aware, pressing CAPS SHIFT and SYMBOL SHIFT simultaneously gives access to a whole new set of functions. Many of these are used by Tasword Two. They are rather a mixed bag and the help page groups them by function.

Extending Operation

Pressing F and G in extended mode gives fast scrolling, in fact scrolling by a screenful at a time, down the text and up the text respectively. (These correspond neatly to F and G pressed together with SYMBOL SHIFT which give scrolling a line at a time in the same direction).

There are four commands to alter formatting. Two of these, controlling right justification and word-wrapping, act as switches, and their state is indicated in the status line. Right justification means a straight right margin achieved by inserting spaces to pad out text to fill lines completely. This is ON unless you specify otherwise. If you decide to change from justified text to non-justified text or vice versa, the STEP command allows you to reformat all your text. Word-wrapping means that, while you are typing, the program automatically judges where a line ends and transfers the whole of the excess word to the next line. The only time the user needs to press ENTER is to force a new line – for example, at the beginning of a paragraph or for a list. You can however over-ride this. You can also opt to justify or unjustify the line the cursor is on.

Typing B and V in extended mode puts in markers to indicate the beginning and end respectively of a block of text. You can then move the normal cursor to indicate the position where you would like either to move or to copy this block of text to. This facility is one that most people would hate to have to do without as it is a feature that allows one to compose text and then change one's mind about it. It is this sort of freedom that makes using a word processor different from using a typewriter.

Sinclair 64

One respect in which we were a little disappointed by Tasword Two is that you are restricted to its predetermined line length of 64 characters per line. There are commands for setting alternative left and right margins but once you have exercised this option you cannot use lots of the program's other facilities. Moreover, once you have reset the margins to normal, you cannot use STEP to reformat any area of text typed in using the altered margin. The only use for the margin commands is to enter short indented passages. When margins are set the 'out of bounds' area of text are shaded which means you see at a glance where the margins are set.

Another three of the extended mode commands are concerned with printing on the ZX Printer. Extended mode P sends the whole text file to the printer, 64 characters to a line. It appears exactly as on the TV screen with one exception. You can select large printing – for titles and headings – by placing the 'large printing on marker' (extended mode L) in front of any portion of the text and the 'large printing off marker' (extended mode K) at its end. When you use

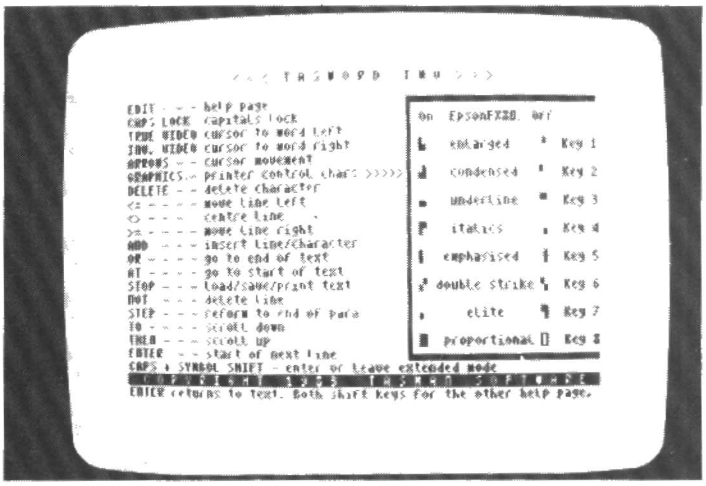
these commands you see a message on the screen but the characters you type appear in their normal size.

As mentioned above, Tasword gives 64 characters to a line. In normal operation you see a 64 column by 22 row screen using specially defined alphabetic characters which are half as wide as the Spectrum's own character set. (Tasman Software also market a machine code utility program to make other programs and output appear with 64 characters per line). If you prefer, you can work with the normal Spectrum characters on a 32 character wide screen. Using the arrow keys you can move the 'window' to the left and right, just as moving up and down the text screen.

Another three of the miscellaneous extended mode commands require a mention. One is the Insert mode switch. The status of the insert mode is indicated at the bottom of the screen. When entering text you will normally want it ON. If however, the text file is almost too big for the available memory you may wish to switch it OFF.

Extended mode X is a dangerous command as it erases all the text in the file. Being user-friendly, however, Tasword only obeys this command after asking "Are you sure".

The replace facility (extended mode R) is a useful one – if rather slow in operation. It enables you to find any single word you specify in the text and replace it by a string, that can include spaces, of up to 32 characters. You might use this facility to save typing out a long title over and over again – for example using "TTWP" when entering text and replacing it by "Tasword Two Word Processor" only when the file was completed – or to correct misspellings.



In Conclusion

The main shortcomings of Tasword Two are limitations of the Spectrum itself. The file size – 320 lines – is rather restricting but not overly so. All this article would fit into it comfortably and to write a longer piece it would just have to be split up. The line count in the status line means you can't easily ignore how much you have written. The fact that the line length is, to all intents and purposes, fixed is perhaps rather a disadvantage but, as Tasword point out, it is a reasonable width for printing on A4 paper (other word processors default to 66 characters per line) and it does fit onto the ZX printer which gives a cheap route to text processing. A well behaved ZX Printer provides output that photocopies beautifully so the idea of using ZX print out is not as silly a notion as at first might appear and if you paste two columns of it on one sheet of A4 you are back to a conventional size.

Applications such as newsletters and school magazines suddenly become very possible propositions with a Tasword Two. Tasword Two certainly does everything it can to get over the shortcomings of the Spectrum's hardware and uses it keyboard facilities with great ingenuity. However, it cannot do much about the difficulties of using the Spectrum's rubber keys. Luckily, the user can do something and Tasword Two persuades me that a "proper" keyboard for the Spectrum is probably a worthwhile investment.

If you're still staggering through the computer jungle and not getting sensible answers to your questions, we have some good news:

Now you can turn to **professional** people who are capable of giving you sound advice on practically all aspects of popular computing.

They all have one thing in common:

They are **COMPUTERS FOR ALL** dealers.

A Computers for All dealer is different from the normal Computer retailer.

Not surprisingly he won't try to sell you things like cameras or cosmetics, stationery or sealing wax.

He will, however, be capable of answering sensibly almost any question you have on computers and computing, and have readily available a wide range of popular computers, hardware, software, books, and peripherals.

So why not call in at your local **COMPUTERS FOR ALL** dealer today? He can lead you in the direction you want to travel.

The shops where people matter:

AVON
MERCATOR COMPUTER SYSTEMS
3 Whiteladies Road, Clifton, Bristol. 0272 731079.

MOBILE MICROS
2 Castle Street, Thornbury, Bristol. 0454 418383.

BERKSHIRE
KENNETH WARD COMPUTERS
Verve House, London Road, Sunningdale. 0990 25025.

CHANNEL ISLANDS
MEGA LTD.
7 Anley Street, St. Helier, Jersey. 0534 72263.

CHESHIRE
THE COMPUTER CENTRE
68 Chestergate, Macclesfield. 0625 618827

CORNWALL
COMPUTAVISION
3 Hill Park Crescent, Saint Austell PL25 5HW. 0726 5297.

FAL-SOFT COMPUTERS
8 St. George's Arcade, Falmouth TR11 3DH. 0326 314663.

DEVON
A & D COMPUTERS
"Computerland", 6 City Arcade, Fore Street, Exeter. 0392 77117.

BITS AND BYTES
44 Fore Street, Ilfracombe, N. Devon EX34 9JD. 0271 62801.

COMPUTER SYSTEMS (TORBAY)
Pump Street, Brixham. 08045 6565/6.

CRYSTAL COMPUTERS
209 Union Street, Torquay. 0803 22699.

SYNTAX LTD.
Midhurst, Grenofen, Tavistock, Devon. 0822 2392.

DORSET
DENSHAM COMPUTERS
329 Ashley Road, Parkstone, Poole. 0202 737493.

ESSEX
AKHTER INSTRUMENTS
Unit 19, Arlinghyde Estates, South Road, Harlow. 0279 412639.

CHELMSFORD COMPUTER CENTRE
3 Baddow Road, Chelmsford. 0245 356834.

COMPUTERAMA
88-90 London Road, Southend-on-Sea, Essex. 0702 335443.

COMPUTERS FOR ALL
72 North Street, Romford. 0708 752862.

HANTS
MICRO VIDEO STUDIOS LTD.
60 Normandy Street, Alton, Hants. 0420 82055.

HERTS
VIDEO CITY
45-47 Fishers Green Road, Stevenage. 0438 53808.

THE COMBINED TRADING CO.
10 & 11 Salisbury Square, Old Hatfield. 07072 65551.

KENT
ANIROG COMPUTERS
29 West Hill, Dartford. 0322 92513.

APHROS SOFTWARE CO.
47 Hawley Square, Margate. 0843 294699.

DATA STORE
6 Chatterton Rd., Bromley, Kent. 01-460 8991.

EWANS COMPUTER CENTRE
17 Burton Road, Kennington, Ashford, Kent. 0233 35826.

MICRO MAGIC
128 Erith Road, Bexleyheath. 0322 523052.

LANCS
P.C.S.
39 Railway Road, Darwen. 0254 776677.

4MAT COMPUTING
67 Friargate, Preston. 0772 561952.

SUMLOCK
Royal London House, 198 Deansgate, Manchester M3 3NE. 061 834 4233.

LEICS
DIMENSION
27-29 High Street, Leicester. 0533 57479.

MOVIES COMPUTER CENTRE
5 Church Street, Melton Mowbray. 0664 61169.

LONDON
EROL COMPUTING
125 High Street, Walthamstow E17. 01-460 8991.

KAYDE HOME COMPUTERS
1 Station Approach, New Eltham, London SE9. 01-859 7505.

KELLY'S COMPUTERMARKET
227 Dartmouth Road, Sydenham, London SE26 4QY. 01-699 4399/6202.

MIDDLESEX
SCREENS MICROCOMPUTERS
6 Main Avenue, Moor Park, Northwood. 09274 20664.

TWILLSTAR COMPUTERS
17 Regina Road, Southall. 01-574 5271.

N. IRELAND
D. V. MARTIN LTD.
13 Bridge Street, Belfast BT1 1LT. 0232 226434.

NORTHAMPTON
LEISURESOFT
13 The Friary, Grosvenor Centre, Northampton. 0604 36726.

RICHARD REEVES
174 Kettering Road, Northampton. 0604 33578.

OXFORDSHIRE
SCIENCE STUDIO
7 Little Clarendon, Oxford OX1 2HP. 0865 54022.

SHETLANDS
TOMORROW'S WORLD
Esplanade, Lerwick, ZE1 0LL. 0595 2145.

SURREY
ANIROG COMPUTERS
8 High Street, Horley. 02934 6083.

COMPUTASOLVE
8 Central Parade, St. Marks Hill, Surbiton. 01-390 5135.

KWX BUSINESS SUPPLIER
Old Coach House, Belmont, Sutton, Surrey. 01-642 9471.

SNOW-BEECH
1 East Grinstead Road, Lingford, Surrey. 0342 832476.

WALES
DAN EVANS (BARRY) LTD.
81 Holton Road, Barry, South Glamorgan. 0446 734242.

MICRO-CARE COMPUTING LTD.
18 Baneswell Rd., Newport, Gwent NPT 4BP. 0633 50482.

STEVE'S COMPUTER CO. LTD.
Castle Arcade, Cardiff, South Glamorgan. 02224 1905.

TRYFAN COMPUTERS LTD.
57 Madoc Street, Llandudno, Gwynedd. 0492 70802.

TYNE & WEAR
THE COMPUSHOP
10 Newgate Centre, Newcastle-upon-Tyne NE1 5RE. 0632 618673.

WARWICKSHIRE
IMPULSE MICRO SYSTEMS LTD.
6 Central Chambers, Cooks Alley, Wood Street, Stratford-Upon-Avon. 0789 295819.

W. MIDLANDS
CALISTO COMPUTERS
119 John Bright Street, Birmingham B1 1BE. 021-632 6458

JBC
200 Earlsdon Ave. North, Earlsdon, Coventry. 0203 73813.

WORCESTERSHIRE
EVESHAM COMPUTER CENTRE
Crown Court Yard, Bridge St., Evesham. 0386 48635.

YORKSHIRE
COM-TEC
6 Eastgate, Barnsley, South Yorkshire. 0226 46972

EMPIRE ELECTRO CENTRE
783-789 Leeds Road, Bradford, W. York. 0274 662476.

TOWERLIGHT LTD.
7 Crown Street, Hebden Bridge, W. York HX7 8EH. 0422 843520.

COMPUTERS FOR ALL DEALERS